A Parameter-Efficient Deep Dense Residual Convolutional Neural Network for Volumetric Brain Tissue Segmentation from Magnetic Resonance Images

Ramesh Basnet

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Applied Science (Electrical and Computer Engineering) at

Concordia University

Montréal, Québec, Canada

August 2020

CONCORDIA UNIVERSITY
SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: Ramesh Basnet

Entitled: A Parameter-Efficient Deep Dense Residual Convolutional Neural Network for Volumetric Brain Tissue Segmentation from Magnetic Resonance Images

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Electrical and Computer Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair
Dr. W.-P. Zhu

_____ External Examiner
Dr. C.-Y. Su (MIAE)

_____ Internal Examiner
Dr. W.-P. Zhu

_____ Co-Supervisor
Dr. M.O. Ahmad

_____ Co-Supervisor
Dr. M.N.S. Swamy

Approved by: _____
Dr. Y.R. Shayan, Chair
Department of Electrical and Computer Engineering

_____ 20___ _____
Dr. Mourad Debbabi, Interim Dean,
Gina Cody School of Engineering and
Computer Science

Abstract

A Parameter-Efficient Deep Dense Residual Convolutional Neural Network for Volumetric Brain Tissue Segmentation from Magnetic Resonance Images

Ramesh Basnet

Brain tissue segmentation is a common medical image processing problem that deals with identifying a region of interest in the human brain from medical scans. It is a fundamental step towards neuroscience research and clinical diagnosis. Magnetic resonance (MR) images are widely used for segmentation in view of their non-invasive acquisition, and high spatial resolution and various contrast information. Accurate segmentation of brain tissues from MR images is very challenging due to the presence of motion artifacts, low signal-to-noise ratio, intensity overlaps, and intra- and inter-subject variability. Convolutional neural networks (CNNs) recently employed for segmentation provide remarkable advantages over the traditional and manual segmentation methods, however, their complex architectures and the large number of parameters make them computationally expensive and difficult to optimize. In this thesis, a novel learning-based algorithm using a three-dimensional deep convolutional neural network is proposed for efficient parameter reduction and compact feature representation to learn end-to-end mapping of T1-weighted (T1w) and/or T2-weighted (T2w) brain MR images to the probability scores of each voxel belonging to the different labels of brain tissues, namely, white matter (WM), gray matter (GM) and cerebrospinal fluid (CSF) for segmentation. The basic idea in the proposed method is to use densely connected convolutional layers and residual skip-connections to increase representation capacity, facilitate better gradient flow, improve learning, and significantly reduce the number of parameters in the network. The network is independently trained on three different loss functions, cross-entropy, dice similarity, and a combination of the two and the results are compared with each other to investigate better loss function for the training. The model has the number of network parameters reduced by a significant amount compared to that of the state-of-the-art methods in brain tissue segmentation. Experiments are performed using the single-modality IBSR18 dataset containing high-resolution T1-weighted MR scans of diverse

age groups, and the multi-modality iSeg-2017 dataset containing T1w and T2w MR scans of infants. It is shown that the proposed method provides the best performance on the test sets of both datasets amongst all the existing deep-learning based methods for brain tissue segmentation using the MR images and achieves competitive performance in the iSeg-2017 challenge with the number of parameters that is 47% to 98% lower than that of the other deep-learning based architectures.

Acknowledgements

I would like to express my sincere gratitude to my thesis supervisors Dr. M. Omair Ahmad and Dr. M.N.S. Swamy for their exceptional support and guidance during my study and research. Without your continuous encouragement and motivation, this work wouldn't have been possible.

I would like to thank my labmates for their wonderful company and time. I greatly appreciate the discussions we had that helped me in my research and making good decisions in difficult situations.

I express my deepest thanks to my family for their unconditional love and support. This work is dedicated to them.

Table of Contents

List of Figures

List of Tables

List of Symbols and Abbreviations

Symbols

α	Momentum
β	Shift parameter
β_1	First order moment decay
β_2	Second order moment decay
ε	Smoothing term
γ	Scale parameter
λ	Learning rate
σ	Standard deviation
ω	Parameter weight

Abbreviations

Adam	Adaptive Momentum
ANN	Artificial Neural Network
ASD	Average Surface Distance
BN	Batch Normalization

CNN Convolutional Neural Network

ConvNet Convolutional Neural Network

CSF Cerebrospinal Fluid

CT Computed Tomography

DenseNet Densely Connected Convolutional Network

DenseVoxNet Densely-Connected Volumetric ConvNet

DSC Dice Similarity Coefficient

EM Expectation Maximization

FA Fractional Anisotropy

FCM Fuzzy C-Means

FCN Fully Convolutional Network

GM Gray Matter

GMM Gaussian Mixture Model

IBSR Internet Brain Segmentation Repository

iSeg 6-month Infant Brain MRI Segmentation

MHD Modified Hausdorff Distance

MRI Magnetic Resonance Imaging

ReLU Rectified Linear Unit

ResNet Residual Network

RF Radio Frequency

SGD Stochastic Gradient Descent

SVM Support Vector Machine

T1w T1-weighted

T2w T2-weighted

VGG Visual Geometry Group

VoxResNet Voxelwise Residual Network

WM White Matter

Chapter 1

Introduction

1.1 Brain Tissue Segmentation

Brain tissue segmentation is an important part of brain image analysis, clinical diagnosis, and neuroscience research. The structural information obtained from the segmentation is substantially used for anatomical analysis, pathological studies, abnormality detection, surgical planning, and image-guided interventions. It has also been used in reparative surgery, radiotherapy, stereotactic neurosurgery, and the study of the correlation between brain anatomy and function [1]. Magnetic resonance imaging (MRI) is profoundly used to obtain the structural information due to its high-resolution, fast non-invasive acquisition and diverse contrast modalities.

Magnetic resonance imaging is a medical imaging technology that is used to produce volumetric scans of anatomical regions of the human body using magnetic fields. It is mainly effective in the areas having high proton or hydrogen density for instance, the regions rich in water molecules. A hydrogen atom has a single proton in the nucleus that possess a magnetic moment. The magnetic moments of the large number of hydrogen atoms in a tissue are typically randomly oriented. When a strong magnetic field is passed through the protons, they align with the applied field and precess around the axis that lie along the direction of the field. The frequency at which the protons precess corresponds to the applied magnetic field that is referred to as resonance frequency. When a radio frequency (RF) pulse with the resonance

frequency is passed through the protons, they transit to higher energy state losing their equilibrium and alter their alignment relative to the applied magnetic field. At the removal of the RF pulse, the excited protons gradually release energy and realign with the applied field. Depending upon the type of tissue i.e. their chemical property or hydrogen density, the amount and rate at which the energy is released differ that distinguishes them from each other and forms the basis of MRI. Although the final MR scans are three-dimensional (3D), during acquisition they are obtained as two-dimensional (2D) slices at fixed intervals over a region of interest.

Usually in an MRI scanner there are three primary components, namely, a primary magnet, a group of three gradient coils and an RF coil. The primary magnet produces the main magnetic field that is applied to the subject. The applied field strength typically ranges from 1.5 to 7 Tesla. The gradient coils produce gradient magnetic fields in three-dimensional space to deform the original magnetic field as a function of position that enables the tracking of the position of the 2D slices in a 3D volume. The RF coil produces the resonance frequency that excites the protons. With the application of the RF pulse, the net longitudinal magnetization of protons parallel in direction to that of the main magnetic field moves to the transverse plane and undergoes 90 degrees phase shift. When the RF pulse is stopped, the net magnetization returns to its initial stage—the process referred to as T1 relaxation. The phase of the transverse component also decays, termed as T2 relaxation. Also, the inhomogeneity in the main magnetic field leads to a faster decay of transverse magnetization, the process called T2* relaxation. The T1 and T2 relaxation times are the foundations of acquisition of the T1w and T2w images that provide contrasting features for tissue distinction. For example, in the case of brain tissues, in T1-weighted images, the cerebrospinal fluid has the darkest intensities and the white matter the brightest, whereas in T2-weighted images, cerebrospinal fluid has the brightest intensities and the white matter the darkest.

Unlike most of the imaging technologies that use harmful ionizing radiations, MRI utilizes magnetic field, which is considered safe for use on living beings, to produce three dimensional anatomical and physiological images of various regions of the body. Its different modalities such as T1-weighted (T1w), T2-weighted (T2w), T1-weighted contrast-enhanced

(T1C), T1-weighted inversion recovery (T1-IR), and T2-weighted fluid-attenuated inversion recovery (FLAIR) provide various contrast information that helps in distinguishing various types of tissue. It has been widely used in the detection of neurodegenerative disorders such as Parkinson's or Alzheimer's diseases, schizophrenia and epilepsy [2–6]. Segmentation of brain tissues into the three labels, white matter (WM), gray matter (GM), and cerebrospinal fluid (CSF) from MR scans provide structural information that assist physiological research, neurological diseases identification and lesion morphometry quantification [4, 7]. For example, gray matter segmentation has been used to study aging by estimating cortical thickness [8] and white matter segmentation from functional MR images has been used to project brain activation maps [9]. Accurate segmentation of these tissues is very sensitive and critical, thus challenging. Although MR images have superior resolution compared to that of many other imaging modalities, in the case of brain tissues, the distribution of intensity in the MR scans is inhomogeneous and largely overlapping. Several artifacts such as subtle movements of the subject during data acquisition, low signal to noise ratio, and presence of more than one tissue in a voxel referred to as partial volume effects further add difficulty to the segmentation task [10]. Manual segmentation of brain tissue from the MR scans is obsolete because of the requirement of a significant amount of human expertise and time to process the large amount of volumetric data. Every subject and voxel need to be treated independently, thus making the process irreproducible. Moreover, manual segmentation, being prone to error due to human lethargy and inter- and intra-observer variability, is not reliable. Thus, an automatic, reliable, fast, accurate, and reproducible method for segmentation of brain tissue is actively pursued. There are several existing brain tissue segmentation methods in the literature that can be broadly grouped into classical, atlas-based, and deep learning-based methods.

Classical method

Image segmentation methods based on intensity, hand-crafted features, boundary detection, region growing, traditional machine learning approaches, or pixel classification can be grouped under the classical method. In these methods, first-order features, namely, indi-

vidual pixel or voxel intensity, and second-order features, namely, the statistically extracted information from the intensities to accommodate local spatial intensity dependence have been abundantly used for medical image segmentation in literature [11–15]. The most basic methods of segmentation perform intensity thresholding on individual pixels. In intensity thresholding methods, the threshold intensity derived from the histogram is individually compared to all the pixels of the input to classify them into different groups. Although this method is fast and efficient, it fails to consider the spatial information of images, is highly dependent on the selection of the threshold [16], and is sensitive to noise and intensity inhomogeneities. Some methods that consider the neighborhood information and are less sensitive to noise than the thresholding methods are region-growing method and clustering. The region growing method groups together the neighboring pixels in a region that have similar properties. A set of initial seed points are selected and the adjacent pixels to the seed points are compared with the seed points based on a similarity criterion, such as the minimum difference between the intensities to classify the adjacent pixels. This method, however, requires manual selection of seed points, is sensitive to the selection of seed points and intensity inhomogeneity, and is affected by noise [17]. Clustering is an unsupervised learning method similar to the region growing method but does not require initial seeds for grouping similar pixels together for classification. This method randomly initializes the initial points. Some popular clustering methods used in image segmentation are k-Means, Fuzzy C-Means (FCM), and expectation-maximization (EM). These methods can also address the intensity inhomogeneity by using iterative correction schemes [18]. Clustering has been popularly used in the segmentation of brain tissues into white matter (WM), gray matter (GM), and cerebrospinal fluid (CSF) [14]. Some statistical and handcrafted feature models such as the Gaussian Mixture Model (GMM) and Support Vector Machine (SVM) have also been applied to the segmentation tasks [19–22]. GMMs model intensity of each tissue to a certain probability distribution, whereas SVMs use spatial and intensity features of the tissues to train the models for classification. These methods represent the data in terms of predefined features which can fail to capture the latent details for distinguishing different types of tissues. All the classical methods rely upon hand-designed features, require extensive

human intervention such as feature extraction, selection or preprocessing, fail to capture inherent information from the input images, are difficult to generalize, or are susceptible to noise, intensity overlaps or inhomogeneities.

Atlas-based method

An atlas is a representative statistical or probabilistic model of an expected anatomical structure or tissue at each pixel or voxel normally constructed by sampling from a population of subjects followed by identification of anatomy or tissue of the sample by segmentation and projection of segmentation to a spatial system [23]. In atlas-based methods, segmentation is done by matching a target image or volume to single or multiple atlases. It is usually performed in three steps—registration where the target image is aligned to single or multiple atlases by transformation or mapping, label propagation where the labels from the atlas are transferred to the target image, and segmentation where the transferred labels are finally identified [24, 25, 13]. Thus, the segmentation task is mainly a registration problem. Except for the construction of atlas, the method is automatic and does not require significant human effort. The registration is usually done is two steps—global and local. Global registration is a simple affine or rigid transformation performed initially to align the target to the atlas. Local registrations follow the global registration. In local registration, non-rigid transformations such as non-linear or free-form grid deformations are applied. Local registration allows a better mapping using complex optimizations at a high computational expense. The use of a single atlas for segmentation is often insufficient to register a broad range of targets and complex deformations may be required to align the targets that are difficult to fit in an atlas. Methods based on the multiple atlas register the target to multiple atlases and based on strategies such as majority voting, weighted voting, expectation maximization, and probabilistic models, combine the registrations to obtain a more accurate and robust segmentation [26]. However, all the atlas-based methods require a priori information, depend largely on the quality of registration, and are subject to errors due to anatomical variability among the targets, tissue deformation due to diseases, etc.

Deep learning method

Deep learning methods, in particular, the Convolutional Neural Networks (CNNs) have been prominently used in the various fields including pattern recognition [27], image classification [28], object detection [29], super-resolution [30], and segmentation [31]. Convolutional neural networks (CNNs) have also been used in the field of medical image segmentation [32] and have shown remarkable improvement in the performance. Compared to other image segmentation approaches, CNNs can learn to abstract relevant features from an image or volumetric scan at different levels without the need for intense preprocessing, extraction of hand-designed features, complex registration, and significant human effort. Different convolutional architectures having 2D, 2.5D and 3D frameworks with various numbers of layers have been employed for medical image analysis including single-modality or multi-modality brain tissue segmentation [32–34]. So far, the deep learning methods are the preferred and state-of-the-art algorithms for brain tissue segmentation. We discuss the deep learning methods used for brain tissue segmentation in Section 1.2.

1.2 Related Works

With the enormous success of convolutional neural networks in image classification and pattern recognition, its application for biological tissue segmentation is currently a prominently researched area of medical image analysis [35]. Initially, two-dimensional CNNs with several layers were used for medical image segmentation of MR images. Zhang et al. [36] utilized multi-modality MR images, namely, T1-weighted (T1w), T2-weighted (T2w), and fractional anisotropy (FA) images to segment infant brain tissues into white matter (WM), gray matter (GM), and cerebrospinal fluid (CSF). The segmentation performance of CNN compared to that of the traditional methods such as random forest (RF), support vector machine (SVM), coupled level sets (CLS), and majority voting (MV) was significantly better in their experiment. They also concluded that using multi-modal data had advantages over using single-modal data as input in a two-dimensional CNN, and that the T1w images had the most discriminating information for segmenting brain tissues into WM, GM, and CSF.

Nie et al. [37] utilized three pathways of a two-dimensional fully convolutional network (FCN) for the three modalities (T1w, T2w, and FA) and combined the feature maps from higher layers of the networks for final segmentation of the brain MR images. In a fully convolutional network [31], the last fully connected layer of the CNN architecture is replaced with a fully convolutional layer that offers better localization performance and pixel-wise predictions of a full-image in a single forward pass. Nie et al. showed that the fusion at higher-level representations of different modalities complemented each other and enhanced the segmentation performance compared to that of the low-level fusion at input. The standard two-dimensional CNNs that use a single slice of the image from three-dimensional MR scans were improved further to utilize the spatial information from 3D planes. Prasoon et al. [38] integrated triplanar 2D CNNs associated with XY, YZ, and ZX planes of 3D MR image for tibial cartilage segmentation from low field knee MR scans. Moeskops et al. [39] also used a single trained instance of 25-layer deep triplanar CNN architecture for the segmentation of six brain tissues in brain MR images, pectoral muscle segmentation in breast MR images, and coronary artery segmentation in cardiac CT angiography (CTA). The use of 3×3 kernels in the convolutional layer of their architecture allowed the structural feasibility of the deeper architecture. The triplanar approach provided superior performance over the 2D approaches and computational efficiency over 3D approaches, however, the use of only three orthogonal planes from the multiple planes is not the best utilization of the available volumetric medical data [40] and can be problematic for CNNs on anisotropic 3D data [41]. A "3D-like" FCN was proposed by Xu et al. [42] to segment volumetric neonatal or adult brain MR images. They stacked three successive 2D slices of brain MR images like a set of 2D color images (R, G, B) accounting for the 3D information to segment the middle slice. Urban et al. [43] introduced one of the first three-dimensional convolutional neural networks for the segmentation of medical images. They utilized a 3D-CNN for brain tumor segmentation using multi-modality MR images. Kamnitsas et al. [40] proposed an 11-layered double-pathway 3D CNN to segment ischemic stroke lesions in the brain using multi-modality MR images. The two parallel paths processed image at multiple scales, instead of a single large-sized 3D input patch, that provided advantages in terms of memory requirement and computation.

Although both pathways had the same sized receptive fields, the input to the second pathway was a patch from a subsampled version of the input image to capture the larger area around a voxel. The architecture was also fully convolutional in nature enabling it to segment larger image (multiple voxels) efficiently in one forward pass. Dolz et al. [44] utilized a 3D fully convolutional network architecture with small kernels to segment subcortical brain using MR images. After the success of U-Net [45] architecture in medical image segmentation, several architectures based on it have been proposed for improving the segmentation performance. Based on FCN, the 23 layered U-Net utilizes contracting and expanding paths to capture local and contextual information with skip connections to transfer high-resolution features at different stages between the contracting and expanding paths. Çiçek et al. [46] replaced all 2D operations of U-Net architecture with their 3D counterparts for biomedical volumetric image segmentation. Milletari et al. proposed V-Net [47], a 3D FCN architecture based on U-Net for segmentation of MR prostate volumes. They utilized residual learning [48] at different stages to improve performance in terms of accuracy and computational time. He et al. [48] reported that increasing the depth of the network to increase the learning capacity subjects the network to a "degradation problem", in that the network performance first gets saturated and then degrades rapidly. To mitigate the degradation problem in the deeper CNN architectures, they introduced residual network (ResNet). VoxResNet [49] also utilized residual networks in a 3D CNN to segment brain MR images. The 25 layered network used multimodality images (T1w, T1-IR, and FLAIR) for segmentation. Wang et al. proposed RP-Net [50] that utilized recursive residual blocks and a pyramid pooling module to segment the brain tissues from volumetric MR images into WM, GM, and CSF. Huang et al. proposed DenseNet [51] that had dense connections to reduce the vanishing-gradient problem, build up feature propagation, encourage feature reuse, and reduce the number of parameters in CNNs. Yu et al. utilized the dense connections in a fully convolutional network, DenseVoxNet [52] to segment the cardiac and vascular structures from 3D cardiac MR images. Bui et al. proposed a skip-connected 3D DenseNet [53] utilizing the advantages of DenseNet, Dolz et al. proposed HyperDenseNet [54] extending dense connectivity to occur between layers of the same path as well as across different paths of multiple modalities,

and Hashemi et. al. [55] used DenseNet with balanced similarity loss functions for infant brain tissue segmentation using volumetric multi-modal brain MR images.

1.3 Motivation

The state-of-the-art deep convolutional neural network architectures used for brain tissue segmentation utilize contracting path to capture contextual information using progressive downsampling layers, expanding path to enable localization through progressive upsampling layers, and skip connections from the contracting path to expanding path to transfer high-resolution information. Depending on the method of combining the contextual features at different scales in the contracting path, these architectures can be grouped into two categories. The architectures in the first category, such as 3D U-Net [46] and 3D-FC DenseNet [55], utilize progressive deconvolution layers gradually upscaling feature maps with a consistent upscaling factor. The feature maps from the contracting path and the expanding path are concatenated via skip-connections. This approach generates large numbers of feature maps at each layer and large numbers of parameters to be learned requiring complex optimization and a large amount of memory. The architectures in the second category, such as VoxResNet [49], 3D-SkipDenseNet [53] and RP-Net [50], upsample feature maps from the contracting path at different scales and concatenate them in the expanding path. This approach can decrease the localization precision thus limiting the representational capacity of the network models. Furthermore, the large-size deconvolution kernels used for multi-scale upsampling also result in large number of parameters. Both the categories have large numbers of parameters that increase memory requirement and computational complexity, and require complex optimization of the parameters.

1.4 Objective

In this thesis, we propose a parameter-efficient three-dimensional deep convolutional neural network for volumetric segmentation of T1w and/or T2w brain MR images into white matter

(WM), gray matter (GM), and cerebrospinal fluid (CSF). The proposed architecture is based on cost-effective use of connections between layers and selection of suitable number and size of kernels to reduce memory requirement, computational complexity, and simplify optimization. The objective of this thesis is to design a novel three-dimensional deep convolutional neural network architecture with reduced number of parameters for compact representation, increased performance, and easier optimization to segment brain tissue from volumetric T1w and/or T2w MR scans into white matter (WM), gray matter (GM), and cerebrospinal fluid (CSF).

The dense connections, popularly used for condensed representation, is employed in the contracting path and residual connectivity between the contracting and expanding paths of the proposed network to facilitate an improved information flow without increasing the number of parameters in the network. As a result, the network facilitates easier optimization, decreased overfitting, improved gradient flow, enhanced representation capacity, and improved performance with a significantly reduced number of parameters.

The proposed method is evaluated utilizing the single-modality and multi-modality datasets containing volumetric brain MR scans of a wide range of age groups—from 6-months old infants to 71 years old adults. For optimization of the proposed network architecture, three different loss functions, namely, cross-entropy loss, dice similarity loss, and a combination of the two are investigated.

1.5 Organization of the Thesis

The thesis is organized as follows.

Chapter 2 presents an overview of different components of CNN used in segmentation. The different layers of CNN, the popular connections between the layers, namely, dense and residual connections, and the methods of training and optimizing the CNNs are discussed. Existing deep-learning based methods used for the brain tissue segmentation are also described in this chapter.

Chapter 3 presents the proposed deep convolutional neural network architecture for volumetric brain tissue segmentation. It provides a detailed description of the network and discusses the utilization of dense and residual connections along with the typical layers of CNN in the proposed network.

Chapter 4 describes the experimental setup for the volumetric brain tissue segmentation using MR images. The datasets used in the experiments are presented. For single-modality brain tissue segmentation, the IBSR18 dataset containing high-resolution T1-weighted MR scans of diverse age groups is utilized. For multi-modality brain tissue segmentation, the iSeg-2017 dataset containing T1w and T2w MR scans of 6-months old infant brains is utilized. The data preprocessing, hyperparameter setup, and the loss function used for the training are presented. Then, the metrics that are used to evaluate the segmentation performance are discussed. Finally, the segmentation performance of the network is presented and a comparison with that of the existing methods of brain tissue segmentation is shown.

Chapter 5 provides the concluding remarks on the problem of segmentation undertaken for investigation in this thesis and suggests possible future work resulting from the scheme proposed in this thesis.

Chapter 2

CNN for Brain Tissue Segmentation

2.1 Introduction

Machine learning is a special subfield of artificial intelligence that aims to provide computers the decision-making capability through inference or by learning patterns from data without explicit instructions or programming. Some early machine learning algorithms have been existing since the 1940s, for example, the model based on biological learning theories by McCulloch and Pitts in 1943 [56], perceptrons in 1958 [57], and systems outperforming humans in tasks like chess-playing in 1997 [58]. Deep learning is an aspect of machine learning that has come to prominence since the last decade, however, the key concepts of deep learning are not new. Back-propagation, an algorithm used to train almost all the present-day deep learning architectures was introduced in 1986 for Artificial Neural Networks (ANNs) which was inspired by the biological neural networks [59]. Convolutional Neural Network (CNN) is a special class of feed-forward deep artificial neural network that has a series of trainable layers performing convolution operations between the input and the output. A modern convolutional neural network showed impressive performance in recognizing handwritten digits in 1989 [27]. It was in 2012, with the availability of a large dataset and advanced computational resources, when a convolutional neural network showed remarkable performance in ImageNet Large Scale Visual Recognition Challenge outperforming the then best performing methods by a significant margin that accelerated the

research in deep learning [28]. Today, convolutional neural networks have applications in diverse fields including computer vision, speech recognition, natural language processing, machine translation, bioinformatics, and medical image analysis.

2.2 CNN Architecture Overview

The basic layers in a convolutional neural network are convolution, pooling, non-linear activation, and fully connected layers. However, recently many layers and connections have been proposed such as batch normalization, dropout, residual connections, and dense connections that enhance the performance of CNNs. The convolutional neural networks such as fully convolutional neural networks used in segmentation tasks that maintain spatial resolution at output employ upsampling layers in the latter part of the network architecture instead of fully-connected layers that are employed in the standard CNNs used for classification or regression. The components of CNNs that are used for segmentation are described as follows.

Convolutional layer

The convolutional layer is the foundational block of a CNN. Although colloquially referred to as convolution, this layer mathematically performs sliding dot product or cross-correlation operation. The sliding operation in a typical convolutional layer is shown in Figure 2.1. The operation is done using a kernel or filter. A kernel is a tensor of learnable values called weights that is slid across and elementwise multiplied to the corresponding elements of an input map. The results of applying a different kernel on each of the input maps are elementwise added together often with a bias term to obtain an output feature map. The process is repeated until a desired number of output feature maps is obtained. In a convolution layer, the number of kernels equals to the number of input maps times the number of required output feature maps. A kernel, usually much smaller in size than the input, has few parameters and can be shared multiple times in a network, thus increasing computational efficiency and reducing memory requirements. It also offers equivariance to translation meaning a change in input results in a

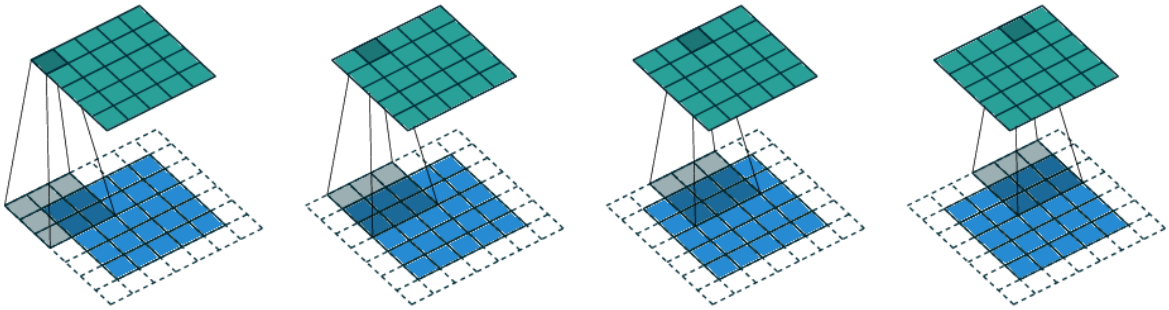


Fig. 2.1 Convolution operation on a 5×5 input using a kernel of size 3×3 with unit padding and unit strides (i.e., $n_{l-1} = 5$, $k_l = 3$, $s_l = 1$ and $p_l = 1$) [60].

change in output in the same way [58]. Let $\boldsymbol{X}_{l-1} \in \mathbb{R}^{H_{l-1} \times W_{l-1} \times D_{l-1}}$ be the input of l^{th} layer, $\boldsymbol{W}_l \in \mathbb{R}^{M \times N \times K}$ be the kernel corresponding to the input layer, $\boldsymbol{b}_l \in \mathbb{R}$ be the bias term, and $\boldsymbol{X}_l \in \mathbb{R}^{H_l \times W_l \times D_l}$ be the output of l^{th} layer. Then, the convolutional operation is defined as:

$$\boldsymbol{X}_l = \boldsymbol{W}_l * \boldsymbol{X}_{l-1} + \boldsymbol{b}_l \tag{2.1}$$

where $*$ represents the sliding dot-product operation. The size of the output can be changed using the parameters of a convolutional layer such as stride and padding. The padding indicates the number of pixels or voxels added to the border of input layer symmetrically whereas stride denotes the steps that the kernel takes while sliding across the input. For example, if $N_{l-1} \times N_{l-1} \times N_{l-1}$ be the size of input, $K_l \times K_l \times K_l$ be the size of kernel, $P_l \times P_l \times P_l$ be the size of padding, and $S_l \times S_l \times S_l$ be the size of stride, then size of output feature map $N_l \times N_l \times N_l$ is given by:

$$N_l = \frac{N_{l-1} - K_l + 2P_l}{S_l} + 1 \tag{2.2}$$

Batch normalization

Initialization and update of weights in the kernels are crucial in a CNN especially in case of deep architectures. Large or small weights might subject a network to exploding and vanishing gradient problems. As data flows through a CNN, it gets multiplied by weights

at each layer and if the weights are small or large, the output will continue getting smaller or larger, respectively, as they move forward through many layers. The similar phenomena happens to the gradients at the backward pass that are correspondingly referred to as exploding and vanishing gradients. Furthermore, large weights might lead the network to an oscillation around minima, or to saturation, for instance, when the network uses sigmoid or tanh activation functions. In the case of small weights, the network will cease to learn as the output and the gradient at every layer diminishes as they move forward and backward through the network, respectively. During the training, the network is also subjected to internal covariate shift [59] where the distribution of input at each layer changes due to the change in network weights. Batch normalization (BN) helps to alleviate these problems of vanishing and exploding gradients, and internal covariate shift by normalizing the input to the layers of the CNN. BN allows each layer of the network to learn more independently of other layers. It reduces the dependence of the network on careful weight initialization and facilitates training at higher learning rates. It also adds a regularization effect and reduces overfitting. Mathematically, if $x_{1...m}$ be the input over the mini-batch B of size m at layer $l-1$, μ_B and σ_B be the mean and the standard deviation of the input data, respectively, then, batch normalized output $y_{1...m}$ at layer l is given by:

$$y_i = \gamma \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}} + \beta \tag{2.3}$$

where γ and β are learnable parameters corresponding to scale and shift, respectively, and ε is a smoothing term that avoids division by zero. At the training, μ_B and σ_B are calculated over the batch as:

$$\mu_B = \frac{1}{m} \sum_{i=1}^{m} x_i \tag{2.4}$$

$$\sigma_B^2 = \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu_B)^2 \tag{2.5}$$

At the testing, an empirical mean and standard deviation over training mini-batches representative of a population is used that can be estimated during training using moving averages.

This makes the output during testing deterministic. The batch normalization algorithm is presented in Algorithm 2.1.

Algorithm 2.1 Batch Normalizing Transform

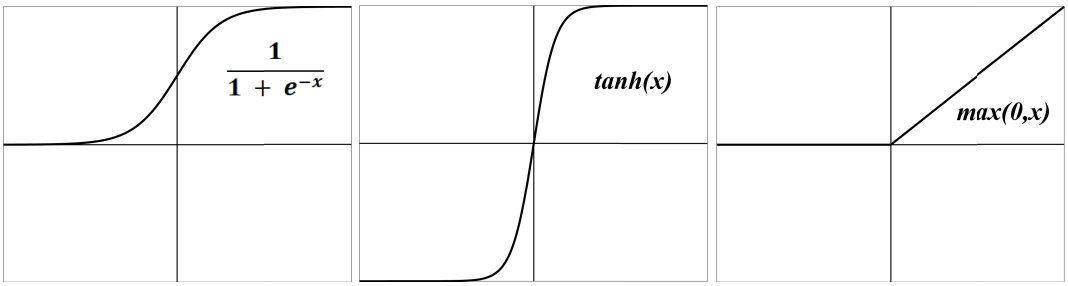
Input: Mini-batch $B = \{x_{1...n}\}$, where n is the number of the input
Output: $\{y_i = BN_{\gamma,\beta(x_i)}\}$; γ and β are parameters to be learned
Calculate the mini-batch mean: $\mu_B \leftarrow \frac{1}{n}\sum_{i=1}^{n} x_i$
Calculate the mini-batch variance: $\sigma_B^2 \leftarrow \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_B)^2$
Normalize: $\hat{x}_i \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2}}$
Scale and Shift: $y_i \leftarrow \gamma\hat{x}_i + \beta \equiv BN_{\gamma,\beta}(x_i)$

Activation layer

The activation layer usually follows the convolutional or batch normalization layer in a CNN architecture. It is a differentiable non-linear function that enables the network model to learn complex mappings between input and output. Non-linearity assures that the output of a layer or layers in a CNN is not merely a linear combination of inputs. The popular activation functions are shown in Figure 2.2. The sigmoid and tanh activations used in the literature have saturation regions where the gradients used in updating the weights for learning approach to zero. This phenomenon leads to a problem called vanishing gradient. The rectified linear unit (ReLU) [61] does not have a saturation region thus is resistant to vanishing gradient problem. Furthermore, ReLU offers faster training of the network compared to the saturating nolinearities [28]. ReLU function maps all the negative inputs to zero and transmits the rest as it is. Therefore, it allows the activation of layers in neural networks to contain one or more true zero values that is called representational sparsity [62]. Mathematically, a ReLU function is given as

$$\boldsymbol{X}_l = max(\mathbf{0}, \boldsymbol{X}_{l-1}) \tag{2.6}$$

where $\boldsymbol{X}_{l-1}$ and $\boldsymbol{X}_l$ are the input and output layers to the ReLU, respectively, and $\mathbf{0}$ is a zero-matrix having the same dimensions as that of input and output.



(a) Sigmoid (b) Tanh (c) ReLU

Fig. 2.2 Activation functions.

Pooling layer

Pooling is a non-linear downsampling method that is used to reduce the spatial dimension of feature maps. It reduces the number of parameters of the network, computational expense, and helps to capture scale-invariant contextual information in the subsequent layers of the network. Usually in pooling, non-overlapping and adjacent square or cubic patches of the input are selected and the maximum (max pooling) or average (average pooling) value of the patch is passed on to the next layer. Max pooling is efficient in capturing important features like edges whereas average pooling extracts the smoothened features. The pooling layer also alleviates noise and clutters in CNNs [63]. Downsampling can also achieved by changing the stride of the convolution in the convolutional layer. For instance, to downsample a three-dimensional feature map by 2, a convolutional layer with kernel size of $2 \times 2 \times 2$ and stride of 2 can be used. Although using convolutional layer for downsampling increases the number of parameters in a network, it can enhance representation capacity and performance of the network [64].

Residual connection

Deeper CNN architectures enhance ability of the network to learn but as the layers increase, the network starts suffering from a "degradation problem" [48]. On increasing the number of stacked layers in a CNN, the ability of the network to learn intricacies of the data and the accuracy also increases until saturation. After that, the performance of the network starts

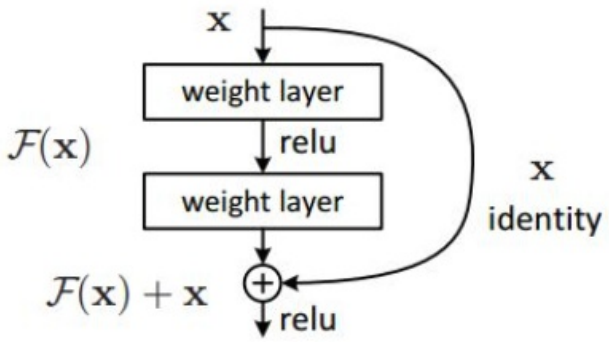


Fig. 2.3 A residual connection between two layers of a convolutional neural network [48].

degrading rapidly leading to higher training error which is called the degradation problem. A residual network was proposed to address this issue in deep architectures. In a residual network, the input to a set of layers is added to the output such that the layers learn a residual mapping between the input and the output. The skip connections that add input to the output as shown in Figure 2.3, commonly referred to as residual connections facilitate information and gradient flow in the network. The input data or the data from earlier layers of the network can be used in the later layers to enable feature propagation. The residual network is computationally efficient as it does not increase the number of parameters of the network and facilitates the design of deeper CNN architectures. Mathematically, a residual connection sums the output of a layer with a skip connection to the output of some previous layer as

$$\boldsymbol{y} = F(x) + x \tag{2.7}$$

where x is the input to a combination of layers, y the output, and F represents the residual mapping to be learned.

Dense connection

Densely connected convolutional network (DenseNet) was proposed to facilitate information and gradient flow between layers of the network, to boost the representational capacity of the

network, and to allow the design of compact network architectures [51]. It enhances feature propagation, reduces feature redundancy, improves gradient flow alleviating the problem of vanishing gradient, and opens a pathway for the efficient and compact architecture design with a smaller number of parameters through feature reuse. In a dense block, an input to a layer is the concatenation of outputs of all the preceding layers through direct connections as shown in Figure 2.4. The collective information from preceding layers increases the variation and complexity in the feature maps of the subsequent layers. As the feature maps can be reused, the network can be deep and condensed, i.e. the layers of the network can have a reduced number of output channels making it easy to train and parameter efficient. Mathematically, dense connections between the layers of a network is defined as

$$\boldsymbol{X}_l = H_l([\boldsymbol{X}_0, \boldsymbol{X}_1, \ldots, \boldsymbol{X}_{l-1}]) \tag{2.8}$$

where $\boldsymbol{X}_l$ is the output of the l^{th} layer, $[\boldsymbol{X}_0, \boldsymbol{X}_1, \ldots, \boldsymbol{X}_{l-1}]$ represents the concatenation output feature maps of all preceding layers and H_l is the l^{th} layer transition. To address the increment in the number of feature maps due to concatenation, a group of densely connected layers is usually followed by a transitional layer. A transitional layer is typically a convolutional layer having kernels of unit size. The convolutional layer is usually preceded by a BN-ReLU (batch normalization followed by rectified linear unit) also known as preactivation unit. The transitional layer also referred to as the bottleneck layer [58], reduces the number of input feature maps thus controlling the increase in number of parameters due to concatenation.

Upsampling layer

CNN architectures that learn pixel to pixel or voxel to voxel mapping such as segmentation of a full image require layers to upscale the low-resolution feature maps to high-resolution. A network can either use interpolation-based upsampling operations such as nearest neighbor, bi-linear or bi-cubic upsamping or transposed convolution operation to upscale the feature maps to a higher resolution. The interpolation-based methods involve mathematical calculations rather than learning. Transposed convolution, also known as fractionally-strided convolution

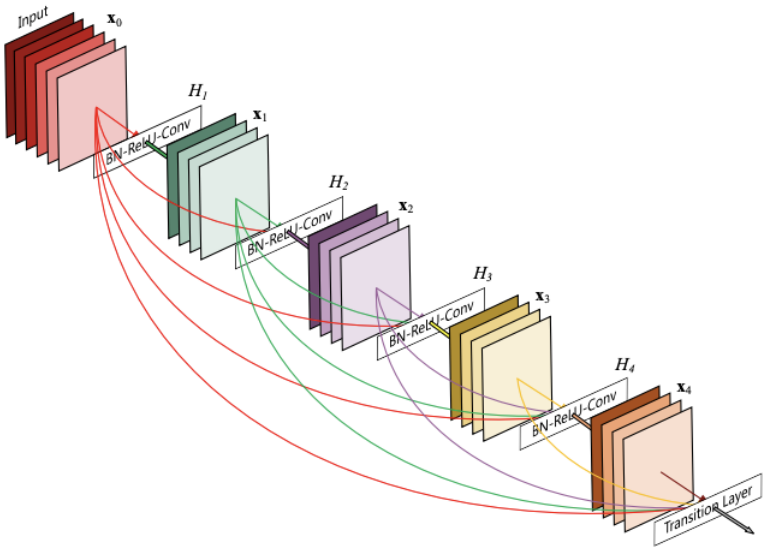


Fig. 2.4 Dense connections between 5 layers of a convolutional neural network [51].

or deconvolution, is a learning-based method to upsample feature maps. It has learnable parameters that can make it an optimal method of upsampling in CNNs. Mathematically, a transposed convolutional layer is similar to a convolutional layer in operation except that each of the element in the input map is padded with a number of zeros specified by a stride before a kernel is applied as shown in Figure 2.5. The output size of the transposed convolutional layer can be calculated as

$$N_l = \begin{cases} N_{l-1} \cdot S_l & \text{if padding = same} \\ N_{l-1} \cdot S_l + max(K_l - S_l, 0) & \text{else} \end{cases} \tag{2.9}$$

where $N_{l-1} \times N_{l-1} \times N_{l-1}$ is the size of the input, $K_l \times K_l \times K_l$ is the size of the kernel, $S_l \times S_l \times S_l$ is the stride, and $N_l \times N_l \times N_l$ is the size of upsampled output feature map.

Dropout

Deep networks that have a large number of parameters has the problem of overfitting [65]. The dropout layer addresses this issue by randomly dropping out or freezing a portion of nodes in a layer during the training. This adds regularization to the network. Moreover,

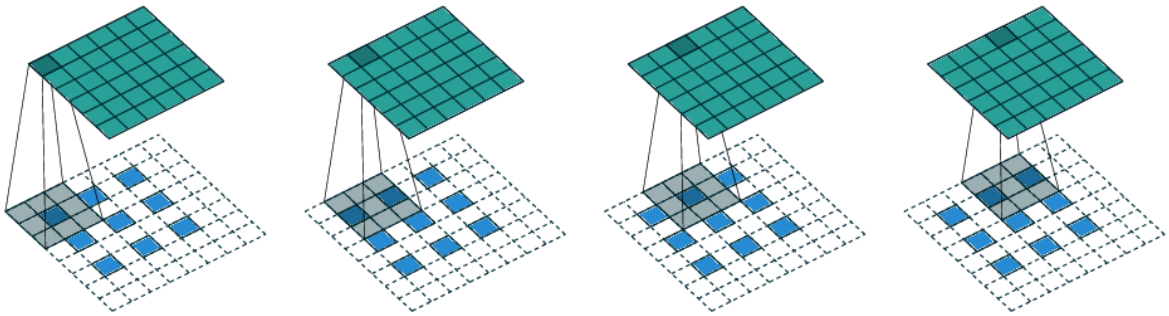


Fig. 2.5 The transpose convolution operation on a 6×6 input using a kernel of size 3×3 with 1×1 zero padding on the border with 2×2 strides (i.e., $n_{l-1} = 6$, $k_l = 3$, $s_l = 2$ and $p_l = 1$). It is equivalent to convolution on a 2×2 input using a kernel of size 3×3 kernel (with 1 zero inserted between inputs) zero padded 1×1 on the border (with an additional border of size 1 added to the bottom and right edges) using unit strides [60].

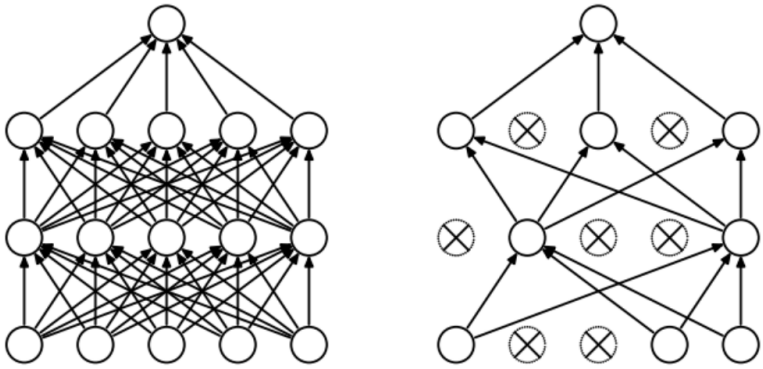


Fig. 2.6 Dropout in a neural network. Crossed units indicate the dropped neurons. [65].

the dropped out units do not participate in the forward and the backward propogation, thus reducing computational complexity during the training. Figure 2.6 illustrates the dropout scheme. A dropout rate, p is a hyperparameter ranging from 0 to 1 that indicates the fraction of nodes that are frozen. During the testing, the dropout is disabled taking all the units of the network into account, and reducing each activation by the factor p.

2.3 Training of CNN

For the CNN architecture to learn the end-to-end mapping from input to the output, the parameters of the network need to be optimized through the training. Training is the process

of updating the parameters of CNN to minimize a criterion that measures how far the obtained outcome is from the expected one. A training process consists of two parts—the forward pass and the backward pass. In the forward pass, the input is passed through the network to obtain a predicted output and the loss is calculated using the obtained output and the expected output or the ground truth. In a backward pass, the gradient of the loss is calculated with respect to the parameters, and through backpropagation algorithm, the parameters are updated such that the loss is reduced. The weights of the network can be updated from gradients using different weight update strategies known as optimization algorithms. Usually, a mini-batch of inputs, which is a collection of multiple inputs, is passed through the network and the average loss of all the inputs in the mini-batch is considered for optimization.

2.3.1 Loss function

In a convolutional neural network, a loss function is a criterion that is maximized or minimized to obtain the optimum network parameters. The gradients of the loss function with respect to the network parameters are used to update the parameters. The choice of the loss function depends on the task the CNN model is designed to accomplish. For segmentation, the popularly used loss functions are cross-entropy loss and dice similarity loss [47].

Cross entropy loss

Cross entropy loss, also called log-loss or softmax loss, measures the difference between a set of predicted probabilities and a set of true or expected probabilities. In a segmentation problem, each of the pixels or voxels in the input data is classified as one of the segmentation labels. The output of segmentation of each voxel is a set of values that is equal in the number to that of the segmentation labels. A softmax function converts the set of output values to a categorical probability distribution consisting of probabilities for all labels that sum to one. Mathematically, softmax function $f_\sigma : \mathbb{R}^M \to \mathbb{R}^M$ is given as

$$f_\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{M} e^{z_j}} \text{ for } i = 1,\ldots,M \text{ and } \mathbf{z} = (z_1,\ldots,z_M) \in \mathbb{R}^M \tag{2.10}$$

The set of probabilities in the categorical distribution gives the likelihoods that a given voxel belongs to the labels. The ground truth probability distribution has probability of 1 for the correct label and 0 for the others. The minimum value of the cross-entropy loss is 0 that is in the case when the output prediction completely matches the expected prediction. From the softmax output and the ground truth probabilities, cross entropy loss is calculated as

$$L_{CE}(Y,\hat{Y}) = -\frac{1}{N}\sum_{b=1}^{N}\sum_{c=1}^{M} Y_{b,c}\cdot log(\hat{Y}_{b,c}) \tag{2.11}$$

where M and N represent the total numbers of labels of segmentation and mini-batch size, respectively, $Y_{b,c}$ is the flattened ground truth and has a value of 1 if the patch b belongs to the label c and 0 otherwise, and $\hat{Y}_{b,c}$ is the flattened predicated probability of the patch b belonging to the label c and has a value between 0 and 1.

Dice similarity loss

The dice similarity coefficient (DSC) measures the degree of overlap between two sets of regions, namely, the predicted and the ground truth regions. DSC ranges from 0 to 1, where 1 means the predicted segmentation perfectly matches the ground truth. Dice similarity loss [47], simply referred to as dice loss, is measured as 1 - DSC or a negative of DSC. Mathematically, dice similarity loss can is given by

$$L_{dice}(Y,\hat{Y}) = -\frac{1}{N}\sum_{b=1}^{N}\sum_{c=1}^{M}\frac{2\cdot Y_{b,c}\cdot\hat{Y}_{b,c}}{Y_{b,c}+\hat{Y}_{b,c}} \tag{2.12}$$

where M and N represent the total numbers of labels of segmentation and mini-batch size, respectively, $Y_{b,c}$ is the flattened ground truth and has a value of 1 if the patch b belongs to the label c and 0 otherwise, and $\hat{Y}_{b,c}$ is the flattened predicated probability of the patch b belonging to the label c and has a value between 0 and 1.

2.3.2 Backpropagation

Backpropagation, often called backprop, is the most widely used algorithm in supervised learning for training the feedforward neural networks. Although it was first introduced in 1960s, it was popularized by Rumelhart, Hinton, and Williams in 1986 [59]. Backpropagation is used to calculate gradient of the loss function with respect to all the learnable parameters of a network model using a simple and computationally inexpensive method of calculus called the chain rule. The chain rule computes the derivative of a function composed of other functions by using the already known derivatives of the composing functions. The gradient of the loss with respect to the parameters of the last layer is calculated first, then the gradients of the parameters of the prior layers are calculated using the already calculated gradient from the subsequent layers and so on. Finally, the calculated gradients of the loss with respect to all the parameters are used to update the parameters using an optimization algorithm. The backpropagation algorithm is described in Algorithm 2.2.

Algorithm 2.2 Backpropagation Algorithm

Input: Mini-batch dataset $B = \{x_{1\ldots M}, y_{1\ldots M}\}$, where M is the mini-batch size
Output: Weights w^l, $l = 1 \ldots L$, where L is the number of layers
  Forward propagate B to calculate x^l at each layer and the loss ℓ
  for $l = L, \ldots, 1$ **do**
    Calculate the derivative of the loss w.r.t to the weights of the layer l: $\frac{\partial \ell}{\partial w^l}$;
    Calculate the derivative of the loss w.r.t to the input of the layer l: $\frac{\partial \ell}{\partial x^l}$;
    Update the weights of the layer l, w^l using an optimization algorithm;
  end for

2.3.3 Optimization algorithm

The gradients of the loss with respect to the network parameters computed from the backprop algorithm is used to adjust the parameters of the network model using an optimization algorithm. An optimization algorithm iteratively updates the learnable network parameters to minimize or maximize the value of loss function such that the predicted and the actual output values are closer to each other. Most of the optimization algorithms that minimize the loss are based on the gradient descent method. If $L(\omega)$ is the objective or loss function at an

iteration η, then the parameter ω is updated using a gradient descent method as follows

$$\omega(\eta+1) = \omega(\eta) - \frac{dL(\eta)}{d\omega(\eta)} \tag{2.13}$$

Gradient descent is based on the idea that if we want to update the parameters to minimize loss, we move down in the direction of the negative gradient towards the optimal parameters. If the loss function needs to be maximized, the process is called gradient ascent, where the steps of update are taken towards the direction of the positive gradient. The most common optimization algorithms that are used in training or updating the parameters of a CNN model to minimize the loss are described below.

Stochastic gradient descent

Stochastic gradient descent (SGD) [27] is one of the basic optimization algorithms to update the parameters in a network. Mathematically, SGD is given as

$$\omega(\eta+1) = \omega(\eta) - \lambda\frac{dL(\eta)}{d\omega(\eta)} \tag{2.14}$$

where λ is the learning rate. Learning rate is a hyperparameter that controls the factor by which the parameters of the network are updated with respect to the gradient of the loss. In other words, it is the step-size of the parameter update. Standard SGD updates parameters using single training data at a time, which is fast but requires frequent updates for optimization and fluctuates quite heavily. Using entire training dataset to compute the loss and update the parameters, known as batch gradient descent, is slow, intractable, and requires significant memory in case of large datasets. Thus, usually a mini-batch gradient descent method is used where a sample of training data called the mini-batch is employed to calculate an average loss for the optimization.

Momentum

Stochastic gradient descent with momentum [66], commonly referred to as momentum, is one of the first sophisticated algorithms based on the SGD algorithm. It calculates the

running sum of the previous gradients and updates the parameter in the direction between the present and the accumulated previous gradient steps. Since the update step doesn't depend only on the current gradient of the loss function but also on the gradient built up over time, the parameter update becomes resistant to off shooting and oscillations of the gradients that leads to an accelerated and stable convergence. Mathematically, parameter update using momentum is done as

$$\Delta\omega(\eta+1) = \alpha\Delta\omega(\eta) - \lambda\frac{dL(\eta)}{d\omega(\eta)} \tag{2.15}$$

$$\omega(\eta+1) = \omega(\eta) + \Delta\omega(\eta+1) \tag{2.16}$$

where λ is a hyperparameter called learning rate, and α is another hyperparameter called momentum set between 0 and 1 that determines the contribution of pervious accumulated gradients and current gradient to the weight update.

Nesterov momentum

Nesterov momentum [67] uses Nesterov accelerated gradient method to update the network parameters. Instead of evaluating the gradient at the current position like the standard momentum, this method takes the momentum step first and then calculates the gradient at the new estimated position as shown in Figure 2.7. Mathematically, the parameter update using Nesterov momentum is done as

$$\Delta\omega(\eta+1) = \alpha\Delta\omega(\eta) - \lambda\frac{dL(\eta)}{d(\omega(\eta)+\alpha\Delta\omega(\eta))} \tag{2.17}$$

$$\omega(\eta+1) = \omega(\eta) + \Delta\omega(\eta+1) \tag{2.18}$$

where λ and α are hyperparamters called learning rate and momentum respectively. In this method, even if the momentum update leads to an overshoot, the gradient update afterwards will redirect it towards the correct direction.

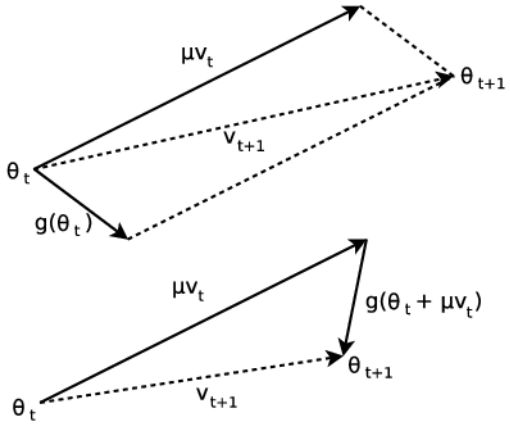


Fig. 2.7 Top: Momentum method, bottom: Nesterov accelerated gradient. where μ is the momentum parameter, same as α in our case [68].

AdaGrad

AdaGrad (adaptive gradient algorithm) [69] stores the running sum of the squared gradients during optimization and divides the learning rate by the square root of the accumulated squared gradients at every iteration. Mathematically, AdaGrad is given as

$$g(\eta+1) = g(\eta) + \frac{dL(\eta)}{d\omega(\eta)}^2 \tag{2.19}$$

$$\omega(\eta+1) = \omega(\eta) - \frac{\lambda}{\sqrt{g(\eta+1)+\varepsilon}} \cdot \frac{dL(\eta)}{d\omega(\eta)} \tag{2.20}$$

where λ is a hyperparameter called learning rate, g is an accumulation variable initially set to zero i.e. $g(0) = 0$, and ε is a smoothing term (usually ranging from 10^{-4} to 10^{-8}) that avoids division by zero. The squared sum of the gradients is less for the small updates that makes the learning rate larger and vice-versa. With AdaGrad, the updates are larger for the infrequent parameters or the parameters with small gradients and are smaller for the frequent parameters or the parameters with large gradients. As a result, the algorithm makes updates at same proportions to all parameters. However, as the number of iterations increase with time or if the initial gradients are large, the sum of the square of gradients keeps getting larger and the learning becomes slow. At the worst case, the learning rate becomes infinitesimally small and the training comes to a standstill.

AdaDelta

AdaDelta (adaptive learning rate method) [70] alleviates the problem of monotonically decreasing learning rates in AdaGrad by limiting the window of accumulated past squared gradients to a fixed size. An exponentially decaying sum of the squared gradients, called the second order moment, is stored and used in adjusting the learning rates. Mathematically, AdaDelta is given as

$$E[g(\eta+1)^2] = \rho E[g(\eta)^2] + (1-\rho)\frac{dL(\eta)}{d\omega(\eta)}^2 \tag{2.21}$$

$$\omega(\eta+1) = \omega(\eta) - \frac{\lambda}{\sqrt{E[g(\eta+1)^2]+\varepsilon}} \cdot \frac{dL(\eta)}{d\omega(\eta)} \tag{2.22}$$

where λ is a hyperparameter called learning rate, ρ is a hyperparameter ranging from 0 to 1, $E[g(\eta)^2]$ is an accumulation variable initially set to zero i.e. $E[g(0)^2] = 0$ and ε is a smoothing term that avoids division by zero.

RMSprop

RMSprop (root mean square propagation) [71] and AdaDelta have been developed independently around the same time to resolve AdaGrad's diminishing learning rates problem. RMSprop is mathematically same as the AdaDelta method. The suggested value by authors of RMSprop for ρ is 0.9 with a learning rate λ of 0.001.

Adam

Adam (adaptive moment estimation) [72] computes the individual learning rates for the network parameters using the first moment (mean) and the second moment (uncentered variance). In addition to calculating the second order moments utilizing exponentially decaying sum of squared gradients, it also uses first order moment that is the exponentially decaying sum of the previous gradients. Thus, the decaying sums of both past and past squared gradients are used to adjust the learning rates. Mathematically, the first and second

moments are calculated as

$$m(\eta+1) = \beta_1 m(\eta) + (1-\beta_1)\frac{dL(\eta)}{d\omega(\eta)} \tag{2.23}$$

$$v(\eta+1) = \beta_2 v(\eta) + (1-\beta_2)\frac{dL(\eta)}{d\omega(\eta)}^2 \tag{2.24}$$

where β_1 and β_2 are the hyperparameters called the decay parameters of first and second moments respectively ranging from 0 to 1 and usually close to 1. However, since the moments are initialized with zeros at the first iteration, they are biased towards zero at initial steps and especially when the decay rates are small. Thus, the bias correction is done by updating the first and second moment as follows

$$\hat{m}(\eta) = \frac{m(\eta)}{1-\beta_1^{\eta}} \tag{2.25}$$

$$\hat{v}(\eta) = \frac{v(\eta)}{1-\beta_2^{\eta}} \tag{2.26}$$

Finally, the parameters of the network are updated as

$$\omega(\eta+1) = \omega(\eta) - \lambda\frac{\hat{m}(\eta+1)}{\sqrt{\hat{v}(\eta+1)+\varepsilon}} \tag{2.27}$$

Algorithm 2.3 describes the adam optimization method.

2.4 Relevent works in Brain Tissue Segmentation

In this section, recent methods based on convolutional neural networks that are used in the brain tissue segmentation are discussed. For segmentation, fully convolutional networks are the foundational methods because of their ability to facilitate voxel-to-voxel mapping from input data to output labels. These architectures encode low-to-high level features at various scales in a series of layers capturing information at global and local contexts. Based on how

Algorithm 2.3 Adam Optimization Algorithm

Input: Objective function $L(\omega)$; initial parameters ω_0, stepsize λ, exponential decay rate for moment estimates $\beta_1, \beta_2 \in [0,1)$, stabilization constant ε

Initialize first moment estimate: $m_0 \leftarrow 0$
Initialize second moment estimate: $v_0 \leftarrow 0$
Initialize iteration step: $\eta \leftarrow 0$
while ω_t has not converged **do**
  Update iteration step: $\eta \leftarrow \eta + 1$
  Computer gradient of objective w.r.t. parameters: $g_\eta \leftarrow \frac{\partial L_{\eta-1}}{\partial \omega_{\eta-1}}$
  Update first moment estimate: $m_\eta \leftarrow \beta_1 \cdot m_{\eta-1} + (1-\beta_1) \cdot g_\eta$
  Update second moment estimate: $v_\eta \leftarrow \beta_2 \cdot v_{\eta-1} + (1-\beta_2) \cdot g_\eta^2$
  Bias-correct first moment estimate: $\hat{m}_\eta = \frac{m_\eta}{1-\beta_1^\eta}$
  Bias-correct second moment estimate: $\hat{v}_\eta = \frac{v_\eta}{1-\beta_2^\eta}$
  Update parameters: $\omega_\eta = \omega_{\eta-1} - \lambda \cdot \frac{\hat{m}_\eta)}{\sqrt{\hat{v}_\eta + \varepsilon}}$
end while

the information is propagated in the network structure for segmentation, we broadly group the architectures into the following categories.

2.4.1 CNN based on contextual fusion

Some popular architectures used in brain tissue segmentation employ multi-level contextual information fusion for localization. The encoder region of these architectures has several downsampling stages to capture contextual information at different scales. However, in the expanding path, the low-resolution features are nonidentically upsampled and concatenated at once.

3D-like FCN

3D-like FCN (3D-like fully convolutional network) [42] adapts layers from a VGG-16 network [73], a popular 16 layered convolutional neural network used on the ImageNet dataset for classification and localization, as the framework of the encoder path. The VGG-16 network stacks 13 convolutional layers with kernels of size 3×3 followed by ReLU activation, has 5 maxpooling layers with kernels of size 2×2 and stride 2 at various stages, and uses

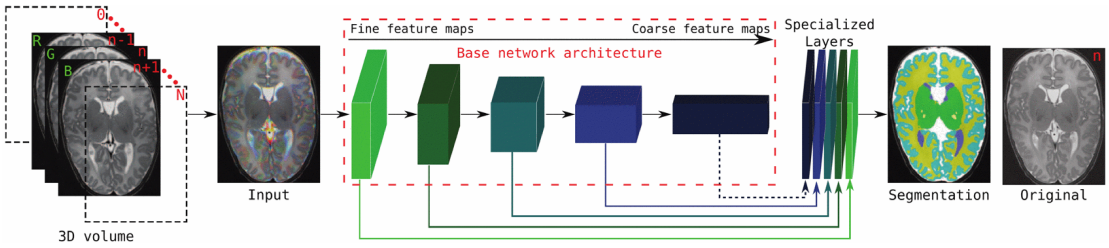


Fig. 2.8 Architecture of the 3D-like FCN. The fine to coarse feature maps of the encoder are combined for segmentation [42].

the 3 fully connected layers at the end of the network for classification. A maxpooling layer is placed after every two convolutional layers in the first two stages and after every three convolutional layers in following three stages of the network architecture. 3D-like FCN disregards the fifth maxpooling layer and the fully connected layers of the VGG-16 network and uses the 5 convolutional stages to capture the contextual information as shown in Figure 2.8. The output of each stage is then passed through specialized convolutional layers with kernels of size 3×3 to reduce the number of feature maps to 16. The outputs of the specialized convolutional layers are then upsampled to the original image resolution and concatenated. Finally, a convolutional layer with kernels of size 1×1 is used to generate the segmentation result from the concatenated feature maps. For the input, the network utilizes three consecutive 2D slices of brain MR scans to segment the middle slice. 3D-like FCN also uses the weights of the VGG network trained on millions of natural images in ImageNet and fine-tunes it for brain tissue classification. This approach is commonly referred to as transfer learning.

VoxResNet

VoxResNet (deep voxelwise residual network) [49] utilizes 3D residual modules inspired from the residual network [48] to extract contextual information for volumetric brain tissue segmentation. VoxResNet has 4 stages in the encoder region separated by pooling layers and a total of 25 layers as shown in Figure 2.9. A residual module in VoxResNet has two sequential convolutional layers whose output is summed to the input elementwise to form a residual representation. The input to the network is passed through two stacked convolutional

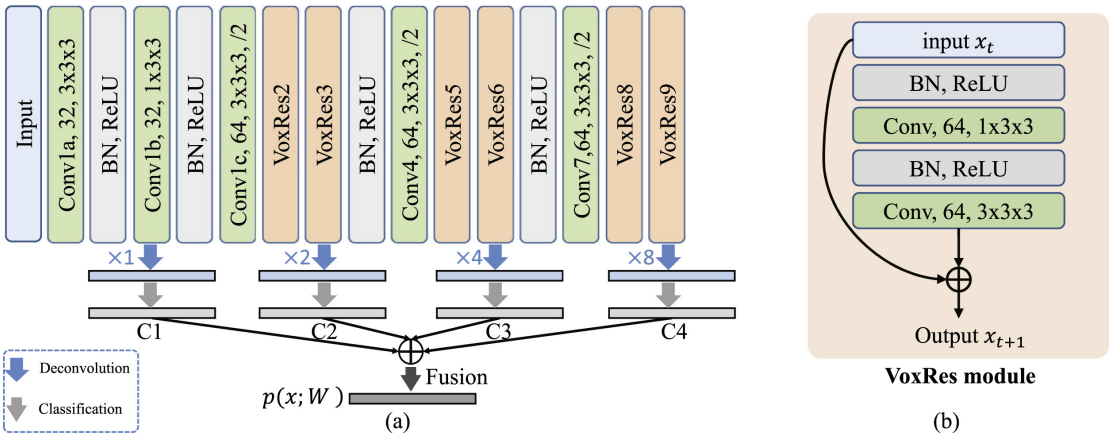


Fig. 2.9 (a) The VoxResNet architecture for volumetric image segmentation. (b) The VoxRes module [49].

layers in the first stage. The subsequent stages have two residual modules stacked together. For pooling, convolutional layer with kernels of size of $3 \times 3 \times 3$ and stride of $2 \times 2 \times 2$ is used. All the convolutional layers in the encoder are followed by BN-ReLU. Except for the first convolutional layer in the residual modules that has kernels of size of $1 \times 3 \times 3$, all the convolutional layers have kernels of size $3 \times 3 \times 3$. The outputs of the 4 stages, taken from the final convolution layers of the stages before the application of BN-ReLU, are passed through 4 auxiliary classifiers for deep supervision [74]. The outputs of the classifiers are then concatenated and passed through the final convolutional layer to produce the output segmentation maps.

3D-SkipDenseSeg

3D-SkipDenseSeg (3D densely connected convolutional network) [53] utilizes densely connected network (DenseNet) in the encoder path for volumetric brain tissue segmentation. With a total of 47 layers, 3D-SkipDenseSeg has 4 stages in the encoder region as shown in Figure 2.10. The first stage has three consecutive convolutional layers with kernels of size $3 \times 3 \times 3$. The remaining stages have dense blocks with pooling layers that gradually downsample the feature resolution to capture contextual information. Each denseblock has 4 pairs of convolutional layers. The two sequential convolutional layers in the pair have kernels of size $1 \times 1 \times 1$ and $3 \times 3 \times 3$, respectively, each preceded by the BN-ReLU unit. Every pair

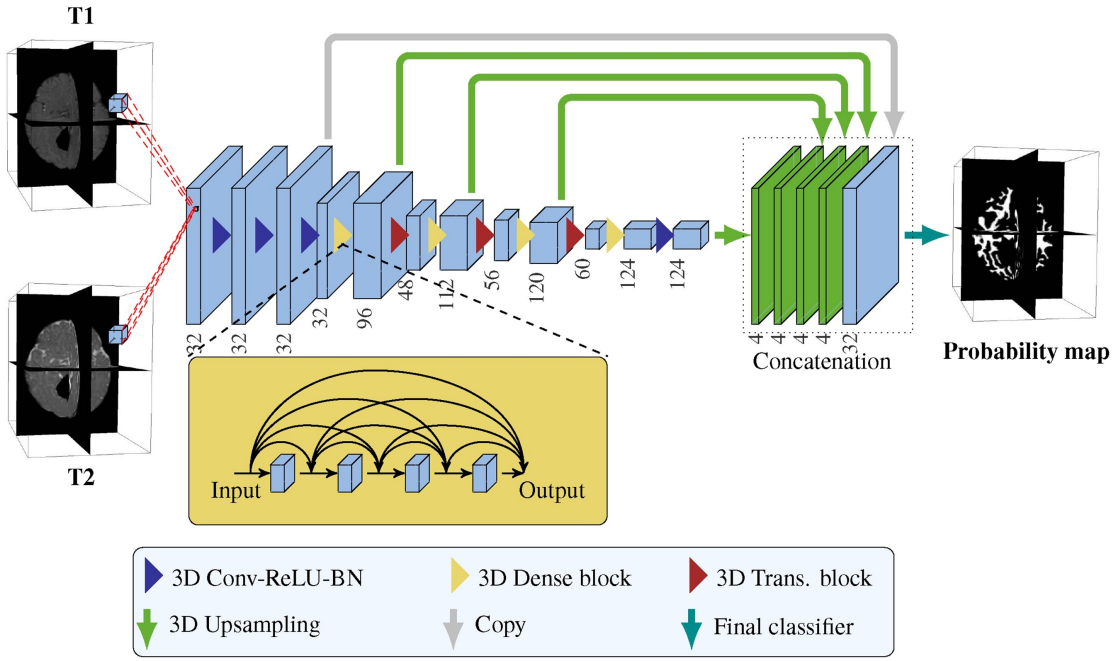


Fig. 2.10 The 3D-SkipDenseSeg architecture for 6-month infant brain tissue segmentation [53].

is densely connected to the preceding pairs. Dropout layers are also used at the end of the denseblocks to reduce overfitting. The pooling at every stage is done using a convolutional layer with kernels of size $3 \times 3 \times 3$ and stride of $2 \times 2 \times 2$. The output at each of the last 4 stages of the encoder path is passed through a convolution layer with kernels of size $1 \times 1 \times 1$ to reduce the number of feature maps and eventually concatenated with the output of the first stage. Finally, a convolutional layer with kernels of size $1 \times 1 \times 1$ is used to generate the output segmentation maps.

2.4.2 CNN based on U-Net

U-Net is one of the most popular CNN architectures used in biomedical image segmentation. The architecture is built upon a fully convolutional network (FCN) architecture where the fully connected layers that are present in the traditional CNNs used for classification are replaced with fully convolutional layers. FCN utilizes upsampled outputs at the latter layer of the network in combination with high-resolution activation maps from earlier layers to extend the use of CNNs from classification to accurate pixel-wise segmentation. U-Net modifies

FCN by adding a large number of feature channels in the upsampling region to effectively propagate the spatial location information. As a result, the contracting and expanding path of the network form a symmetric structure that makes the framework look like a u-shaped architecture. U-Net makes effective use of local and contextual information. Usually in CNNs, the use of small-size patches to facilitate localization fails to capture contextual information of the input whereas, the use of large-sized patches with subsequent pooling layers to facilitate contextualization consequently reduces the localization accuracy. U-Net addresses this tradeoff by utilizing skip connections between the contracting and expanding paths. The contracting path of U-Net captures the contextual information using convolutional layers with stage-wise pooling layers and the shortcut skip-connections transfer local spatial information from the contracting path to the expanding path by combining the high-resolution local features of the contracting path with the upsampled features of the same resolution in the expanding path. The U-Net has 4 stages of different resolutions in the contracting as well as the expanding path as shown in Figure 2.11. It has a total of 23 layers and 7,759,521 parameters. The contracting path has two sequential convolutional layers with kernels of size 3×3 followed by a maxpooling layer with pooling size 2×2 and stride of 2 in each stage. At every stage, the number of feature maps are doubled after downsampling. The contracting path is then connected to the expanding path by two consecutive convolutional layers with kernels of size 3×3 each. The expanding path has 4 transposed convolutional layers with kernels of size 2×2 for upsampling. Each upsampler doubles the resolution of feature maps while reducing the number of feature maps by one-half, and is followed by two consecutive convolutional layers, both of them having kernels of size 3×3. Finally, a convolutional layer with kernels of size 1×1 followed by a sigmoid activation function is used to obtain the predicted segmentation map. All convolutional layers in the network that have the kernels of size 3×3 are followed by a ReLU activation. The feature maps before the pooling layer at each stage of the contracting path are concatenated to the corresponding output feature maps of upsampler in the expanding path having the same resolution using a skip-connection. After the success of U-Net, architectures based on it were proposed for improved performance in medical image segmentation. The network structure of some of the

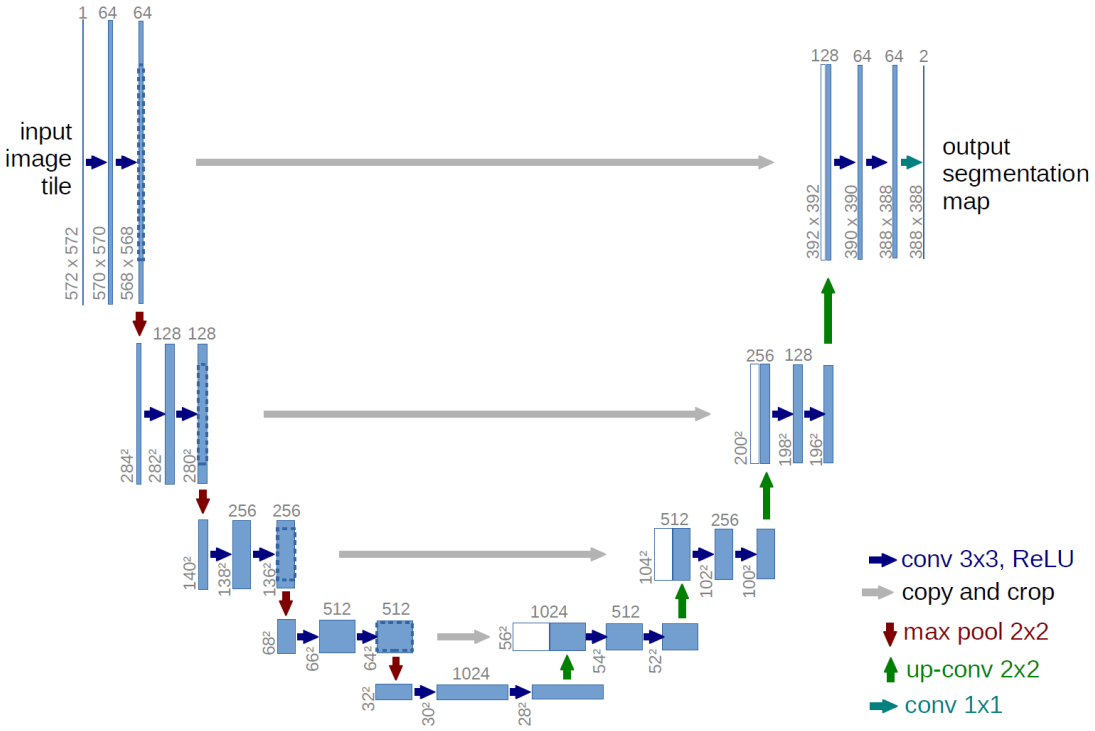


Fig. 2.11 The U-Net architecture. The blue box corresponds to a multi-channel feature map and the white box represents copied feature maps. The number of channels is presented on top of the box. The lower left edge of the box shows the feature resolution and the arrows indicate the operations [45].

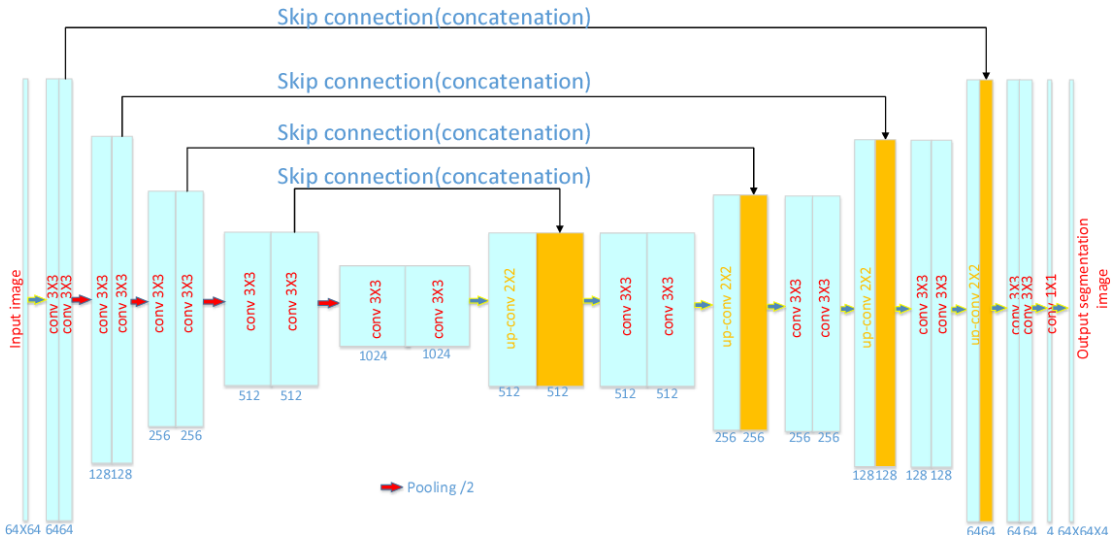


Fig. 2.12 The modified U-Net architecture (2D) for single-modality brain tissue segmentation [75].

popular architectures based on U-Net used in brain tissue segmentation are briefly discussed below.

Modified U-Net

The Modified U-Net (2D) [75] architecture is very similar to the U-Net architecture as shown in Figure 2.12. The modified U-Net was used for single-modality brain tissue segmentation. It segmented brain tissues into three labels at once. It also avoided bottlenecks [76] by doubling the number of feature maps before pooling. The input size to the network was 64×64 that alleviated the memory requirements and the output size was $64 \times 64 \times 4$ for segmenting the input into background and three brain tissue labels. By using padding, modified U-Net maintained the resolution of input at output.

U-SegNet

U-SegNet [77] is a hybrid architecture of SegNet and U-Net as shown in Figure 2.13 that was used for the segmentation of brain tissue into WM, GM and CSF from MR images. SegNet [78], unlike U-Net, uses pooling indices to upsample feature maps instead of transpose convolutional layers. The skip-connections like the U-Net is used in SegNet architecture to

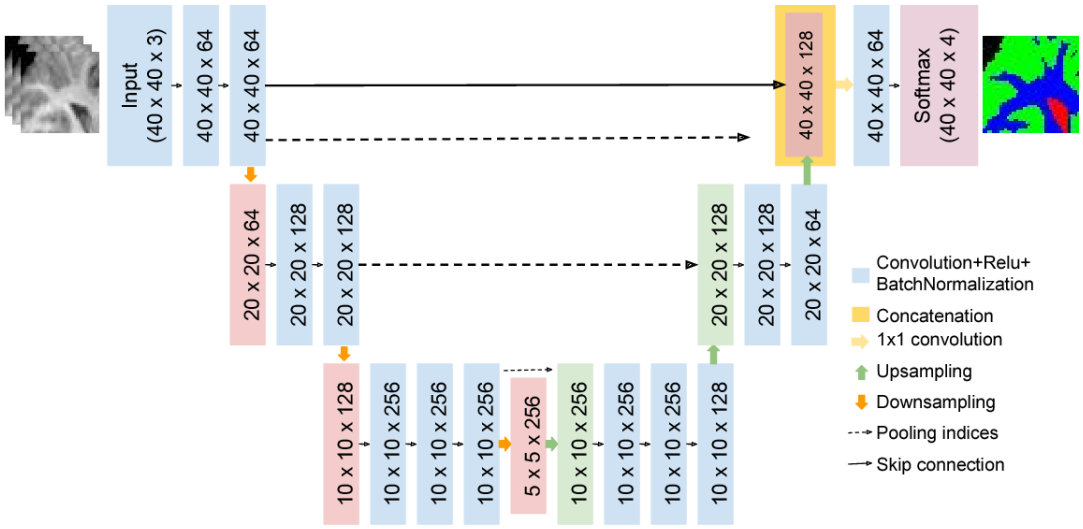


Fig. 2.13 U-SegNet architecture—a hybrid of two popular deep learning segmentation architectures SegNet and U-Net, for brain tissue segmentation [77].

form U-SegNet. U-SegNet utilizes patches of input instead of full image for segmentation by postulating that local information has more importance over contextual information in tissue segmentation. Furthermore, in U-SetNet, the depth of the SegNet architecture is reduced to accommodate the input patches of size $40 \times 40 \times 3$ into computational memory.

3D-Unet

3D U-Net [46] was proposed for volumetric segmentation by replacing all 2D operations of the U-Net architecture by 3D. The 3D-Unet architecture is illustrated in Figure 2.11. To reduce the number of parameters, the stages of the architecture were reduced to 3 resulting in a total of 18 layers. Furthermore, the number of kernels was doubled before the pooling layers to avoid bottlenecks [76]. However, it still has a significant increase in the number of parameters (19,078,593 parameters) and the required huge memory.

RP-Net

RP-Net [50] utilizes residual connections through recursive residual blocks in a U-Net like architecture. Figure 2.15(c) illustrates the structure of RP-Net architecture. The 3D architecture has 4 stages in the contracting path, each consisting of a recursive residual block

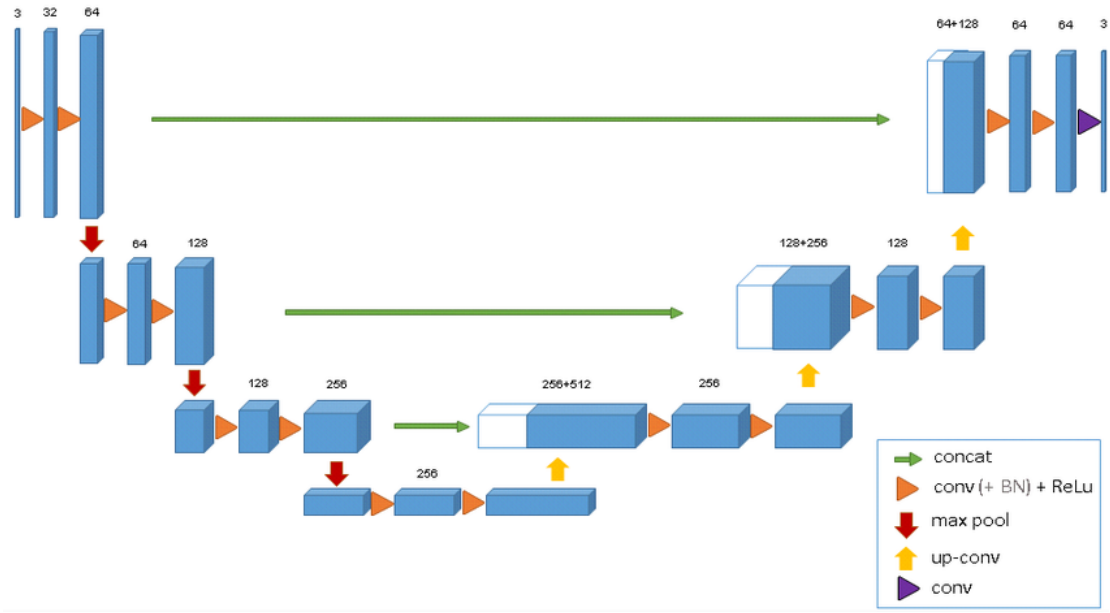
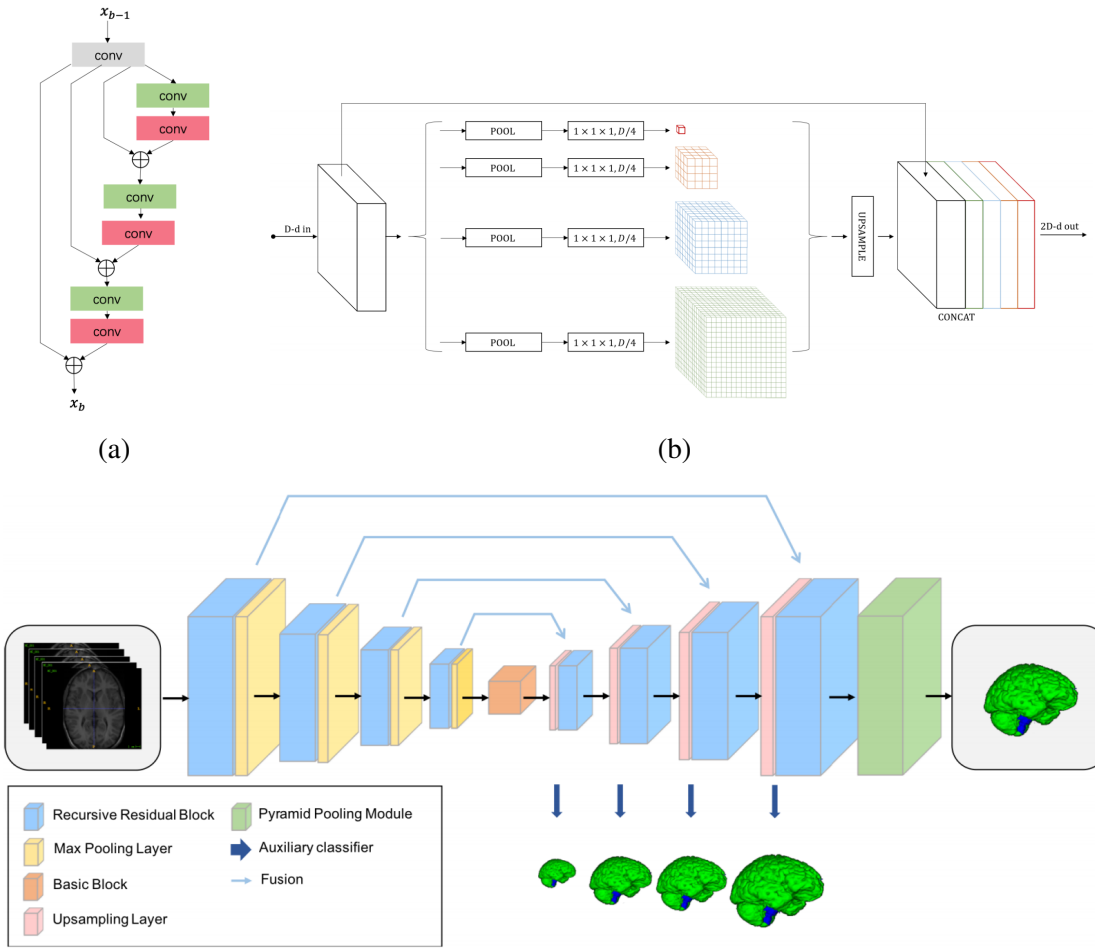


Fig. 2.14 The 3D-UNet architecture. The blue boxes represent the feature maps and the number of channels is presented above the feature maps. [46].

as shown in Figure 2.15(a) having 3 residual units. The stages are separated by maxpooling layers having kernels of size $2 \times 2 \times 2$ and stride of 2. In the expanding path, the upsampling layers are also followed by the recursive residual blocks. All the convolutional layers use kernels of size $3 \times 3 \times 3$ and the skip-connection between the contracting and expanding paths perform element-wise summation. At the end of the network architecture, a pyramid pooling module shown in Figure 2.15(b) is used to effectively capture various levels of contextual information for final segmentation. The pyramid pooling module upsamples the feature maps at different scales and concatenates them similar to that in the contextual fusion strategy. Furthermore, the output of the summation at each stage is passed through a convolutional layer having kernels of size $1 \times 1 \times 1$ in the expanding path to perform deep supervision [74] using an auxiliary loss function.

3D FC-DenseNet

FC-DenseNet [79] (fully convolutional densenet) builds upon the idea of DenseNet for semantic segmentation. It has contracting and expanding paths like U-Net, however, the group of stacked convolutional layers in U-Net at every stage of contracting and expanding



(a) (b)

(c)

Fig. 2.15 (a) A recursive residual block with 3 residual units. (b) Pyramid pooling module. (c) The architecture of RP-Net. The boxes and arrows denote the different operations. The orange box represents a basic block containing two convolution layers each followed by a BN-ReLU unit [50].

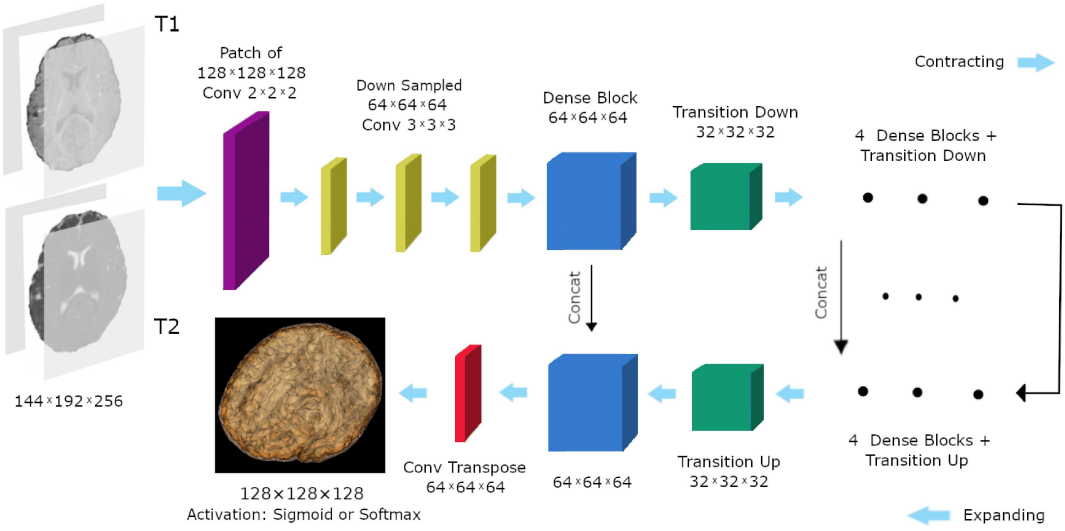


Fig. 2.16 The 3D FC-DenseNet architecture. The purple and red blocks represent the first convolutional downsampling block and last transpose convolutional upsampling block respectively [55].

paths is replaced with a group of densely connected convolutional layers called a denseblock. In between the two denseblocks in the contracting path, there is a downsampling layer that reduces the feature resolution by one-half whereas in between the two denseblocks in the expanding path, there is a upsampling layer that doubles the feature resolution. Skip-connections are used to facilitate transfer of high-resolution local features from the contracting path to the expanding path between the layers of same resolutions. Combining the FC-DenseNet and 3D-SkipDenseSeg architectures, Hashemi et al. proposed a deep three-dimensional architecture for brain tissue segmentation called 3D FC-DenseNet [55] as shown in Figure 2.16. The input to the architecture is are sub-volumes of size $128 \times 128 \times 128$ that are downsampled in the first layer to the size of $64 \times 64 \times 64$ using a convolutional layer with kernels of size $2 \times 2 \times 2$ with stride 2. It utilizes 5 skip connections at multiple stages to transfer the high-resolution feature information from contracting path to the expanding path through concatenation. Finally, a transposed convolutional layer with kernels of size $2 \times 2 \times 2$ and stride 2 is used to upsample the feature maps from $64 \times 64 \times 64$ to $128 \times 128 \times 128$.

2.4.3 CNN based on DenseNet

Some architectures used in the segmentation of brain tissues from MRI do not employ the skip connections or stepwise contracting path to encode contextual information for segmentation like the architectures described in Section 2.4.1 and Section 2.4.2. These architectures, however, utilize the dense connections to increase the information flow and representative capacity for segmentation.

DenseVoxNet

DenseVoxNet (Densely-Connected Volumetric ConvNet) [52] utilizes 2 denseblocks in a network separated by a transitional downsampling layer for segmentation as shown in Figure 2.17. A convolutional layer with kernels of size $3 \times 3 \times 3$ and stride of 2 is placed at the beginning of the network to reduce the feature resolution by one-half. In DenseVoxNet, each denseblock has 12 convolutional layers with kernels of size $3 \times 3 \times 3$ where each convolutional layer is preceded by a BN-ReLU. The transitional layer consists of a BN-ReLU followed by a convolutional layer with kernels of size $1 \times 1 \times 1$ and a maxpooling layer that is a convolutional layer with kernels of size $2 \times 2 \times 2$ and stride 2. Finally, the output of the second dense block is passed through a convolutional layer that is preceded by BN-ReLU and two transpose convolutional layers both having kernels of size $2 \times 2 \times 2$ to match the input resolution. Finally, a convolutional layer with kernels of size $1 \times 1 \times 1$ is used to obtain the segmentation map. The network also uses a dropout layer after every convolutional layer to reduce overfitting. Furthermore, DenseVoxNet utilizes deep supervision [74] by upsampling the output of the convolutional layer in the transitional region using a transpose convolutional layer with kernels of size $2 \times 2 \times 2$.

HyperDenseNet

HyperDenseNet [54] utilizes 3D dense connections in a multi-modality setting without pooling layers. The architecture of HyperDenseNet is shown in Figure 2.18. Usually, two paths are set up for two modalities of input scans. Besides the dense connection in each

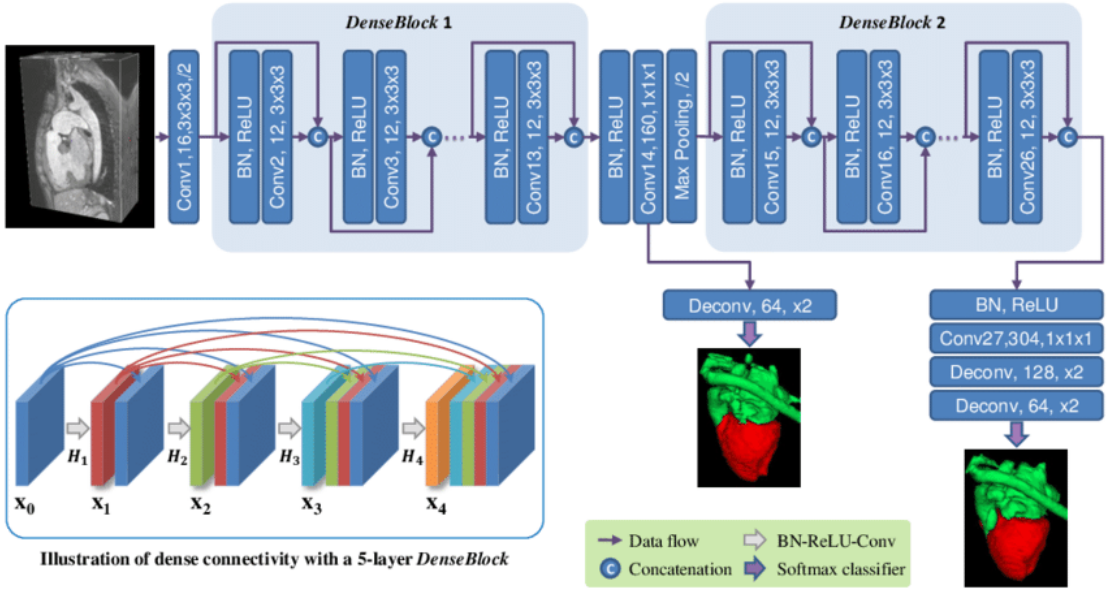


Fig. 2.17 The DenseVoxNet architecture. The bottom-left graph illustrates a 5-layer dense-block as an example. [52].

path, the connectivity is also shared between the two paths. As a result, the input to each convolutional block is a concatenation of feature maps of all preceding layers from both paths. The architecture uses sub-volumes of size $27 \times 27 \times 27$ for training and non-overlapping sub-volumes of size $35 \times 35 \times 35$ for inference.

2.5 Discussion on the Methods

Convolutional neural networks have been very popular in brain tissue segmentation. These networks have been improved for better performance over time by introducing different types of layers, various types of connections between the layers, defining loss functions suitable for specific problems, and modifying optimization methods for faster and better convergence. The fully convolutional networks (FCN) have become a popular method for segmentation problem as it facilitates the voxel-wise classification of input data. By replacing the the fully connected layers of typical CNNs used in classification by fully convolutional layers, the local information is preserved in the FCN. However, contextual information is as important as local information for a network to utilize the semantic information in the data

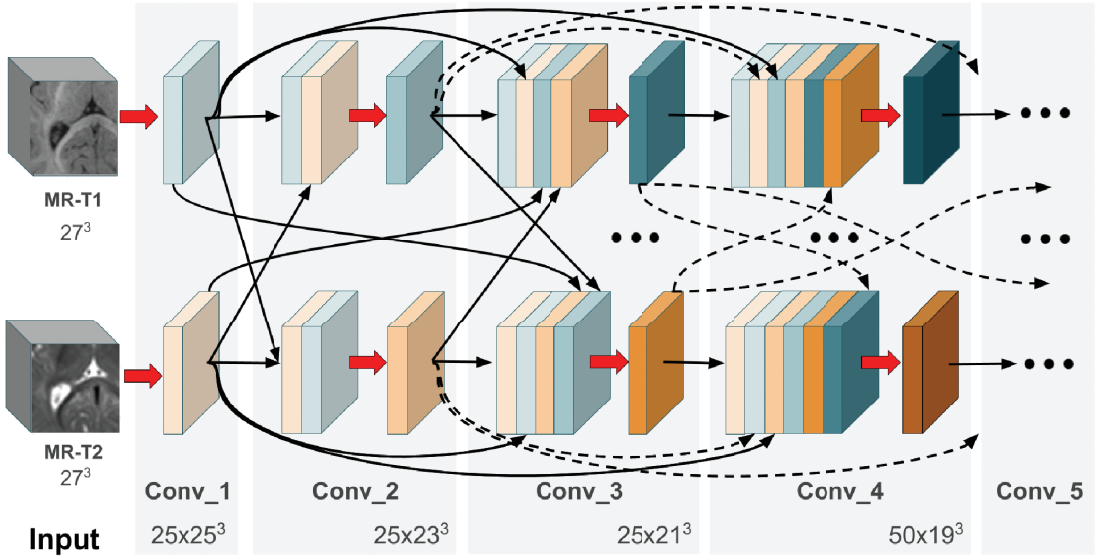


Fig. 2.18 A section of the proposed HyperDenseNet for two image modalities. Gray regions represent the convolutional blocks. Convolution operations and the dense connections between feature maps are denoted by red and black arrows respectively. [54].

for a robust segmentation. U-Net-like architectures that are built upon FCNs have typical layers of CNNs such as convolutional layers, pooling layers, and upsampling layers, and the skip connections to utilize contextual and local information for improved segmentation performance. Based on the contextual information utilization, multi-scale fusion strategies and the architectures built upon U-Net were introduced to make the network more robust. Similarly, the use of volumetric data as input is preferred over the 2D slices in segmentation to maximize the capture of the spatial contextual information for better performance. In general, increasing the depth of the network should increase the representational capacity of the network but in case of deep networks, it also leads the network to a degradation problem where the performance of the network decreases severely. To boost the representational power of the network in deeper architectures while addressing the degradation problem, ResNet was introduced that has shortcut connections allowing efficient gradient flow between the network layers. DenseNet introduced dense connections for better information propagation among the layers and efficient utilization of network parameters. Employing residual or dense connections in the networks based on U-Net or multi-scale fusion strategies have enhanced the performance in brain tissue segmentation. When designing a robust network

for segmentation, the challenge is to utilize the parameters efficiently and increase the representational capacity with as less complexity as possible. Complex architectures with a large number of parameters make the network model difficult to optimize, increase the changes of overfitting, and require a large amount of memory.

2.6 Summary

In this chapter, the components that build up a convolution neural network, the process of training the convolutional neural networks, and the different types of deep learning architectures used in brain tissue segmentation have been discussed. Then, the structures of these architectures have been presented and the common problems in the networks such as requirement of complex optimization, increased memory, and computational complexity due to the large number of parameters have been discussed. In the next chapter, the methods used for addressing these issues in a deep convolution neural network will be discussed and a parameter-efficient convolutional neural network for brain tissue segmentation will be proposed.

Chapter 3

Design of Proposed Architecture

3.1 Introduction

Convolutional neural networks are currently the preferred methods for the segmentation of brain tissues from MR images. CNNs are efficient at abstracting high and low-level representations of data by learning important features at various layers of abstraction. Without the need of manual feature extraction and human supervision, CNNs can be employed on a huge amount of data for various purposes including classification and segmentation. The effectiveness of CNNs has increased over time through the design of complex architectures having different numbers, position, and connection of layers, hyperparameters selection, optimization techniques, and regularization methods. In brain tissue segmentation using MR images, architectures based on the contextual fusion strategies, built upon U-Net, or based on dense connectivity without contextual information utilization have achieved state-of-the-art performances. The existing architectures based on contextual fusion strategies that employ multi-scale upsampling of feature maps at once might not have better localization precision especially when large-sized transpose convolution kernels are used. The large-sized kernels also result in a large number of parameters. The existing architectures built upon U-Net for brain tissue segmentation utilize feature concatenation with the large number of feature maps at each layer leading to a large number of parameters to be learned, thus increasing optimization complexity and memory requirement. The existing architectures

utilizing just the dense connectivity fail to utilize the contextual and semantic information present in the input data. These architectures require multiple kernel operations at each stage increasing the computational complexity. Thus, in general, all the existing architectures used in volumetric brain tissue segmentation require large memory, increased computational complexity, and complex optimization of the parameters. We propose a three-dimensional deep convolutional neural network architecture for volumetric segmentation of T1- and/or T2-weighted MR images of brain into white matter (WM), gray matter (GM), and cerebrospinal fluid (CSF) that addresses the issues of memory requirement, computational complexity, and optimization difficulty in the existing network architectures. The proposed network architecture is built upon the U-Net and utilizes residual skip connections between the contracting and expanding paths to facilitate an improved gradient flow with a significantly reduced number of parameters, thus leading to easier optimization, decreased overfitting, increased representation capacity, and improved performance compared to the architectures in the literature for volumetric brain tissue segmentation. The proposed architecture also utilizes dense connectivity in the contracting path for compact representation by effectively reusing the network parameters without sacrificing the segmentation performance.

3.2 Architecture

The proposed architecture is built upon the U-Net framework along with the use of dense and residual connectivities as shown in Figure 3.1(b) [80]. Like the U-Net architecture, the proposed network has contracting and expanding paths for contextualization and localization. The contracting path in the proposed architecture has a series of denseblocks that capture multi-level contextual information at different scales and resolutions. Each denseblock has a group of layers connected to each other through dense connections. These connections also allow the design of a compact architecture through feature reuse. The network also has residual skip connections to transfer the high-resolution features for localization from the contracting path to the expanding path between the layers of the same resolution without

increasing the number of parameters. We discuss the primary structures that built up the proposed network architecture as follows.

3.2.1 Denseblock

The denseblock is the foundational structure in the contracting path of the proposed network architecture. It mainly consists of convolutional layers with dense connections. It is designed to capture maximum information from the input at various levels of abstractions with a smaller number of parameters. The ability of dense connections to reuse the features from all the preceding layers allows the design of a compact network architecture. Figure 3.1(a) illustrates the different components of the denseblock used in the proposed network. The denseblock is preceded by a downsampling layer that is followed by 4 pairs of convolutional layers. The input to each pair is the collection of outputs of all previous pairs and the input resulting into the dense connections. The downsampling layer is a convolutional layer having kernels of size $2 \times 2 \times 2$ and stride of 2 that reduces the resolution of the input by one-half. Each pair in the denseblock has a convolutional layer with kernels of size $1 \times 1 \times 1$ followed by a convolutional layer with kernels of size $3 \times 3 \times 3$. The small size of kernels in the convolutional layers allows the network architecture to go deeper while being cost-effective. The small receptive field captures local and complex details in the volumetric scans. As the features from previous layers are concatenated, there is an uncontrolled increase in the number of feature maps as we move forward through the network. If k_0 be the number of input feature maps to the dense block then the number of output feature maps at l^{th} layer is given by $k \times (l-1) + k_0$ where k is the output of the convolutional layer before concatenation commonly known as the growth rate. We use a growth rate of 16 in the denseblocks that is also the number of output feature maps of the second convolutional layer in the pair. A small growth rate is feasible in DenseNet because of the dense connectivity that allows collective feature propagation from the preceding layers. This also makes the denseblock parameter efficient. The first layer in the convolutional pair referred to as bottleneck layer [51] limits the number of features going through the second layer to further reduce computational expense as the large number of parameters that would be required to process the concatenated input

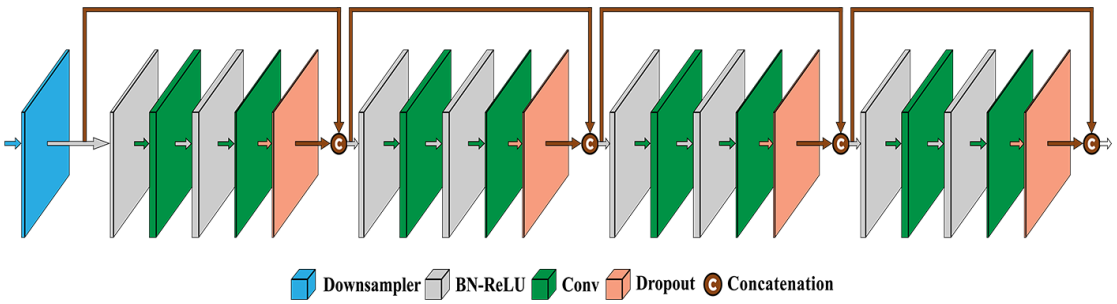
maps is reduced by the bottleneck layer. The number of output feature maps of the bottleneck layer in the denseblock is 64. Every convolutional layer in the denseblock is preceded by a batch normalization and rectified linear unit, also known as the pre-activation unit. The position of the pre-activation unit before convolutional layers have shown better performance in deeper architectures [81]. A dropout layer with the dropout rate of 0.2 is used at the end of the denseblock to reduce overfitting and facilitate regularization.

3.2.2 Transitional block

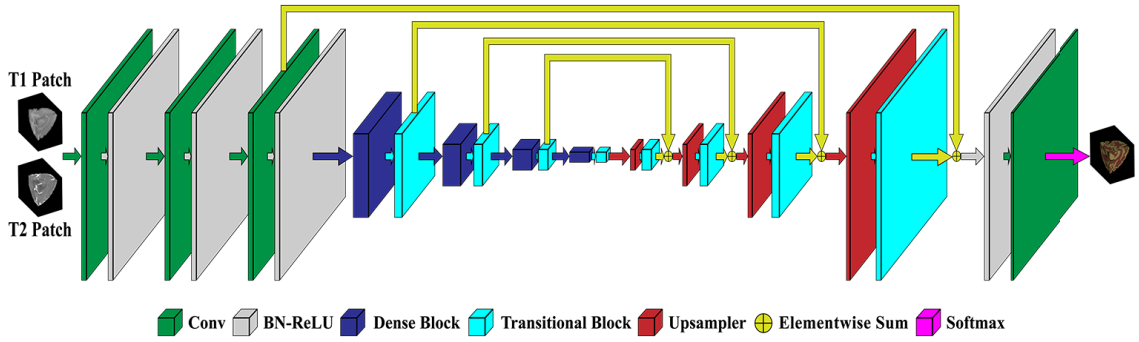
Transitional block adds more representational capability to the network while maintaining the number of feature maps for parameter efficiency and facilitating the information transfer between the layers of the network. We use transitional blocks in the contracting as well as the expanding paths of the proposed network. The transitional block consists of a preactivation unit (BN-ReLU) followed by a convolutional layer having kernels of size $1 \times 1 \times 1$. In the contracting path, the transitional block follows the denseblock and reduces the number of feature maps from the denseblocks by one-half, thus reducing the network complexity and size. In the expanding path, the transitional layer follows the upsampler i.e. the transpose convolution layer. Here, it increases the number of feature maps to capture and propagate the contextual information between the layers and to match the number of feature maps between the layers of the same resolutions in the contracting and expanding paths that are connected through the residual skip connections.

3.2.3 The proposed network

The diagram of the proposed network architecture is shown in Figure 3.1(b). It has two parts, a contracting path and a corresponding expanding path. The network has 706,934 parameters and 52 layers. The contracting path has a total of 43 layers with 4 denseblocks each followed by a transitional block and the expanding path has a total of 9 layers with 4 corresponding upsamplers each followed by a transitional block. The corresponding transitional blocks in the contracting and expanding paths having the same resolutions are connected using



(a)



(b)

Fig. 3.1 (a) The denseblock with downsampler used in the contracting path of the proposed network architecture. The downsampler layer (blue) reduces the feature resolution by one-half. (b) The proposed network architecture for single-modality and multi-modality volumetric brain tissue segmentation. The thickness and size (height and width) of the blocks represent the relative depth and resolution of the feature maps, respectively. The arrows indicate the direction of feature propagation.

the residual skip connections. The contracting path of the proposed network starts with three sequential convolutional layers with kernels size $3 \times 3 \times 3$ to capture low-level features from the input data. Each convolutional layer is followed by a pre-activation unit i.e. batch normalization (BN) followed by a rectified linear unit (ReLU), and outputs 32 feature maps. These convolutional layers are followed by 4 consecutive dense blocks (Section 3.2.1) that encode the high-level features at different scales. The denseblock is followed by a transitional block (Section 3.2.2). A pre-activation unit precedes each of the downsampling layers in the contracting path. The expanding path has 4 transpose convolutional layers or upsamplers with a consistent upscaling factor to upsample the feature maps. Each upsampler doubles the feature resolution and reduces the number of feature maps by one-half. The use of small-sized kernels with consistent upsampling in stages reduces the number of parameters by a significant amount compared to that in multi-scale upsampling using kernels of small to large sizes. The upsampler is followed by a transitional layer that increases the representational capacity of the network and facilitates element-wise summation between the corresponding layers of the same resolutions of the contracting and expanding paths using residual skip connections. The residual skip connections allow effective information transfer from the contracting path to the expanding path without an increase in the number of parameters. All the convolutional layers in the network use zero padding to maintain the feature resolution. Finally, a convolutional layer with kernels of size $1 \times 1 \times 1$ preceded by a pre-activation unit is used to obtain 4 feature maps corresponding to the four labels of the brain tissue segmentation, viz., WM, GM, CSF and the background. Each of the resulting feature maps is subjected to a softmax function to obtain the probability that a voxel of the input data belongs to a label of the brain tissue segmentation. Table 3.1 and Table 3.2 show the detailed description of the denseblock and the proposed network architecture.

3.3 Summary

In this chapter, first, the motivation behind the design of a parameter-efficient convolutional neural network for brain tissue segmentation has been discussed. Next, the design of the

Table 3.1 The denseblock for growthrate 16 and x number of input feature maps.

Layer	Number of input channels	Number of output channels	Number of parameters
BN + Conv3D ($1 \times 1 \times 1$)	x	64	$66x$
BN + Conv3D ($3 \times 3 \times 3$)	64	16	27776
BN + Conv3D ($1 \times 1 \times 1$)	$x+16$	64	$66(x+16)$
BN + Conv3D ($3 \times 3 \times 3$)	64	16	27776
BN + Conv3D ($1 \times 1 \times 1$)	$x+32$	64	$66(x+32)$
BN + Conv3D ($3 \times 3 \times 3$)	64	16	27776
BN + Conv3D ($1 \times 1 \times 1$)	$x+48$	64	$66(x+48)$
BN + Conv3D ($3 \times 3 \times 3$)	64	16	27776
DenseBlock (8 layers)	x	$x+64$	$264x+117440$

proposed three-dimensional deep convolutional neural network architecture with reduced number of parameters for compact representation, increased performance, and easier optimization has been presented. Finally, the components used in the architecture have been discussed and the network structure presented in detail. In the next chapter, the experimental setup for utilizing the proposed network for segmentation of brain tissue is presented and experiments are performed to evaluate the performance of the network and its performance is compared with that of the existing architectures.

Table 3.2 Details of the proposed network architecture.

Block	Layer	Number of input channels	Number of output channels	Number of parameters
Initial Block	Conv3D $(3 \times 3 \times 3)$ + BN	2	32	1792
	Conv3D $(3 \times 3 \times 3)$ + BN	32	32	27712
	Conv3D $(3 \times 3 \times 3)$ + BN	32	32	27712
Downsampler1	Conv3D $(2 \times 2 \times 2,\ s2)$	32	32	8192
DenseBlock1	$8 \times$ (BN + Conv3D)	32	96	125888
TransBlock1	BN + Conv3D $(1 \times 1 \times 1)$	96	48	4800
Downsampler2	BN + Conv3D$_{+Bias}$ $(2 \times 2 \times 2,\ s2)$	48	48	18576
DenseBlock2	$8 \times$ (BN + Conv3D)	48	112	130112
TransBlock2	BN + Conv3D $(1 \times 1 \times 1)$	112	56	6496
Downsampler3	BN + Conv3D$_{+Bias}$ $(2 \times 2 \times 2,\ s2)$	56	56	25256
DenseBlock3	$8 \times$ (BN + Conv3D)	56	120	132224
TransBlock3	BN + Conv3D $(1 \times 1 \times 1)$	120	60	7440
Downsampler4	BN + Conv3D$_{+Bias}$ $(2 \times 2 \times 2,\ s2)$	60	60	28980
DenseBlock4	$8 \times$ (BN + Conv3D)	60	124	133280
TransBlock4	BN + Conv3D $(1 \times 1 \times 1)$	124	62	7936
Upsampler1	ConvT3D $(4 \times 4 \times 4,\ s2)$	62	31	3968
TransBlock5	BN + Conv3D $(1 \times 1 \times 1)$	31	60	1922
Upsampler2	ConvT3D $(4 \times 4 \times 4,\ s2)$	60	30	3840
TransBlock6	BN + Conv3D $(1 \times 1 \times 1)$	30	56	1740
Upsampler3	ConvT3D $(4 \times 4 \times 4,\ s2)$	56	28	3584
TransBlock7	BN + Conv3D $(1 \times 1 \times 1)$	28	48	1400
Upsampler4	ConvT3D $(4 \times 4 \times 4,\ s2)$	48	24	3072
TransBlock7	BN + Conv3D $(1 \times 1 \times 1)$	24	32	816
Final Block	BN + Conv3D$_{+Bias}$ $(1 \times 1 \times 1)$	32	4	196
Network	52	2	4	706934

Chapter 4

Experiments and Results

In this chapter, the experimental setup of the proposed network for volumetric brain tissue segmentation is presented, the experiments are performed, and the results are analysed. The chapter is organized as follows. First, the descriptions of the two datasets used in the experiments are presented. Next, the preprocessing steps carried out on the dataset before passing the data through the network for training or testing is discussed. Then, the method of initializing the parameters of the network is discussed and the training scheme with the hyperparameters setup and the loss functions used for training the network are presented. The metrics that are used to evaluate the finally, performance are discussed and subsequently the segmentation performance of the proposed architecture and a comparison with that of the existing methods of brain tissue segmentation is carried out.

4.1 Datasets

We evaluate the performance of the proposed network architecture on two publicly available datasets containing volumetric MR scans, namely, Internet Brain Segmentation Repository (IBSR18) and 6-month infant brain MRI Segmentation (iSeg-2017). We perform single-modality brain tissue segmentation using the T1w brain MR images of IBSR18 dataset and multi-modality brain tissue segmentation using the T1w and T2w brain MR images of iSeg-2017 dataset into white matter (WM), gray matter (GM), and cerebrospinal fluid (CSF).

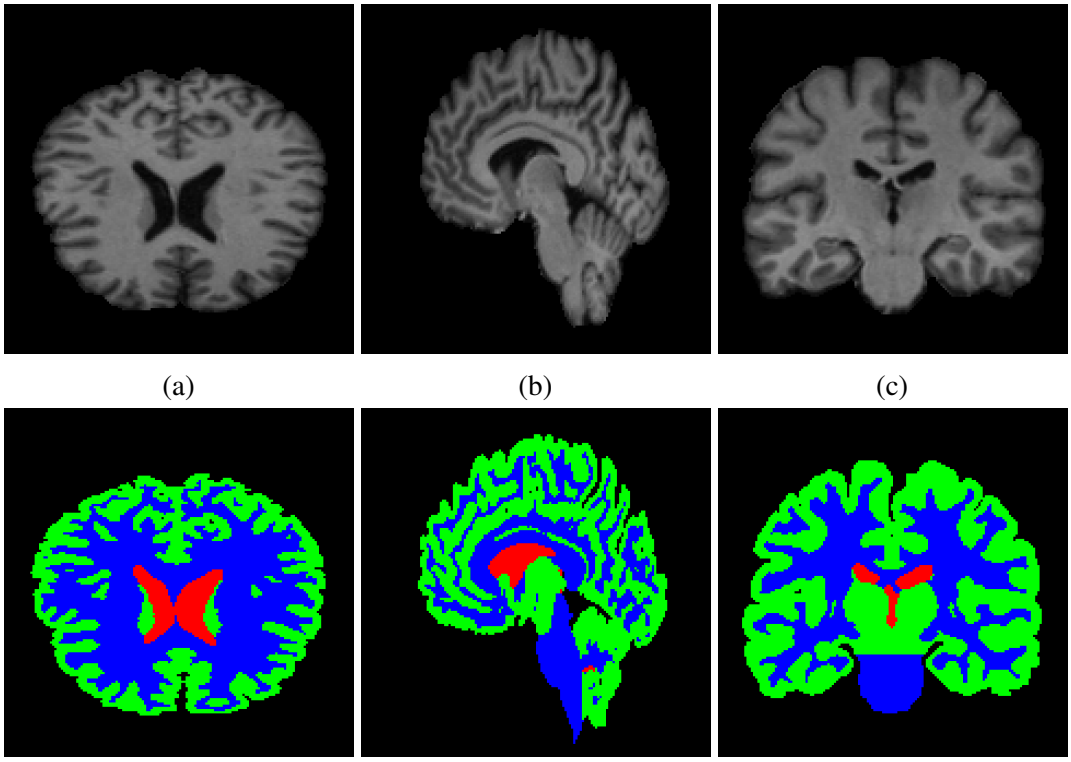
4.1.1 IBSR

The Internet Brain Segmentation Repository (IBSR18) dataset[1] has a collection of T1-weighted (T1w) volumetric brain MR scans of 18 subjects with manual segmentation from experts [82]. The dataset with manual segmentations was provided by the Center for Morphometric Analysis at Massachusetts General Hospital and is available at http://www.cma.mgh.harvard.edu/ibsr/. The subjects belong to the age groups ranging from 7 to 71 years old. Figure 4.1 shows a sample of T1w scans of the IBSR18 dataset along three axes with manual segmentation. For our experiments, we use the latest IBSR V2.0 dataset that contains subjects of volumetric size $256 \times 256 \times 128$ with voxel resolution of $0.84 \times 0.84 \times 1.5\ mm^3$, $0.94 \times 0.94 \times 1.5\ mm^3$ or $1.0 \times 1.0 \times 1.5\ mm^3$. Table 4.1 presents the details of all the subjects of IBSR18 dataset. The "segTRI_ana" file of each subject inside the root folder "IBSR_V2.0 skull-stripped NIFTI" contains the manual segmentation of the brain tissue into WM, GM, CSF, and background. The MR scans have already been skull-stripped, positionally normalized, and bias-field corrected. We use this dataset for single-modality volumetric brain tissue segmentation.

4.1.2 iSeg

The iSeg-2017 dataset[2] is released for the MICCAI Grand Challenge on 6-month infant brain MRI Segmentation [83]. The dataset contains T1-weighted (T1w) and T2-weighted (T2w) volumetric brain MR scans of 10 infant subjects for training and 13 for testing. All the scans are randomly chosen from the Multi-visit Advanced Pediatric (MAP) Brain Imaging Study[3]. Figure 4.2 shows a sample of T1w and T2w scans of the iSeg-2017 dataset with manual segmentation. The training dataset containing 10 subjects contains manual segmentation of every voxel into WM, GM, CSF, and background, whereas the manual segmentation of 13 subjects of testing dataset is not publicly released. The T1w scans were procured with 144 sagittal slices at the resolution of $1 \times 1 \times 1\ mm^3$, flip angle of 7°, and

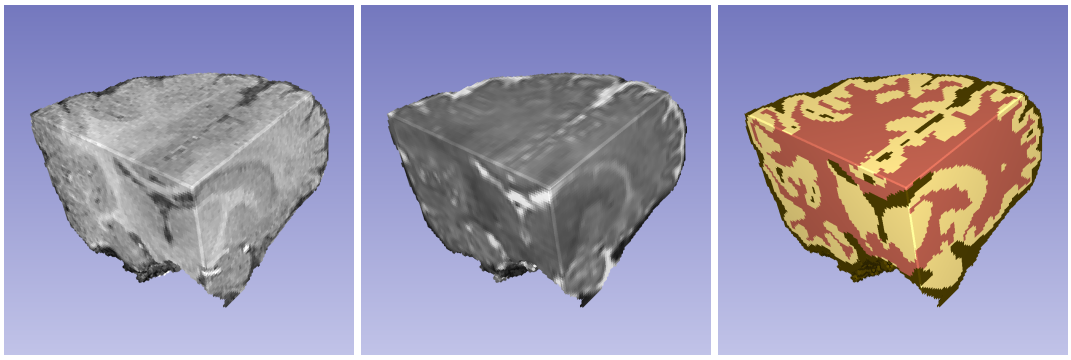
[1] https://www.nitrc.org/frs/?group_id=48&release_id=2316
[2] http://iseg2017.web.unc.edu/download/
[3] http://iseg2017.web.unc.edu/baby-connectome-project/



(a) (b) (c)

(d) (e) (f)

Fig. 4.1 (a) Axial, (b) Sagittal, (c) Coronal slices of T1-weighted MR scan, and (d) Axial, (e) Sagittal, and (f) Coronal slices of manual segmentation (CSF (Red), GM (Green), and WM (Blue)) of IBSR18 dataset (Subject 9).



(a) (b) (c)

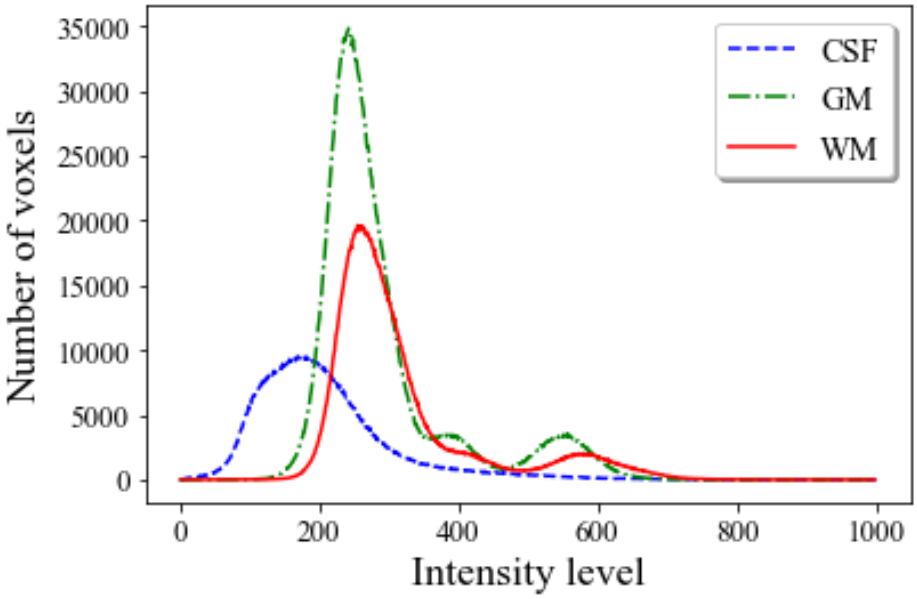
Fig. 4.2 Subvolumes of (a) T1-weighted MR scan, (b) T2-weighted MR scan, (c) Manual Segmentation—CSF (Brown), GM (Yellow), and WM (Orange) of subject 9 of iSeg-2017 dataset.

Table 4.1 The IBSR18 dataset

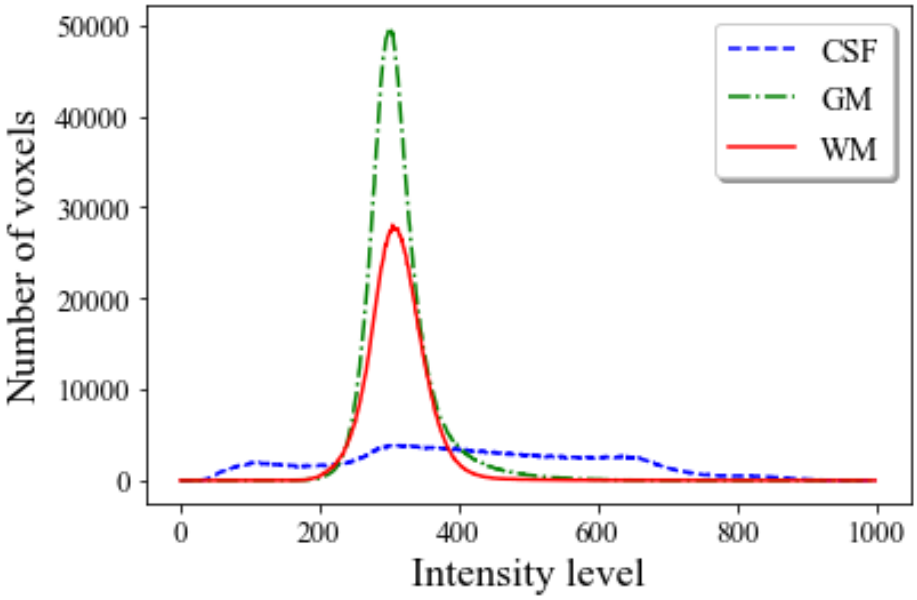
Volume Number	Age (Sex)	Volume Size	Resolution (mm^3)
01	37 (M)	$256 \times 256 \times 128$	$0.94 \times 0.94 \times 1.5$
02	41 (M)	$256 \times 256 \times 128$	$0.94 \times 0.94 \times 1.5$
03	JUV (F)	$256 \times 256 \times 128$	$0.94 \times 0.94 \times 1.5$
04	JUV (M)	$256 \times 256 \times 128$	$0.94 \times 0.94 \times 1.5$
05	41 (M)	$256 \times 256 \times 128$	$0.94 \times 0.94 \times 1.5$
06	46 (M)	$256 \times 256 \times 128$	$0.94 \times 0.94 \times 1.5$
07	70 (F)	$256 \times 256 \times 128$	$1.0 \times 1.0 \times 1.5$
08	60 (M)	$256 \times 256 \times 128$	$1.0 \times 1.0 \times 1.5$
09	41 (M)	$256 \times 256 \times 128$	$1.0 \times 1.0 \times 1.5$
10	35 (F)	$256 \times 256 \times 128$	$1.0 \times 1.0 \times 1.5$
11	59 (F)	$256 \times 256 \times 128$	$1.0 \times 1.0 \times 1.5$
12	71 (M)	$256 \times 256 \times 128$	$1.0 \times 1.0 \times 1.5$
13	JUV (M)	$256 \times 256 \times 128$	$0.94 \times 0.94 \times 1.5$
14	JUV (M)	$256 \times 256 \times 128$	$0.94 \times 0.94 \times 1.5$
15	8 (M)	$256 \times 256 \times 128$	$0.84 \times 0.84 \times 1.5$
16	7 (M)	$256 \times 256 \times 128$	$0.84 \times 0.84 \times 1.5$
17	8 (M)	$256 \times 256 \times 128$	$0.84 \times 0.84 \times 1.5$
18	13 (M)	$256 \times 256 \times 128$	$0.84 \times 0.84 \times 1.5$

TR/TE ratio of 1900/4.38 ms. The T2w scans were acquired with 64 axial slices at the resolution of $1.25 \times 1.25 \times 1.95$ mm^3, flip angle of 150°, and TR/TE ratio of 7380/119 ms. The scans were realigned to the corresponding T1w scans and resampled to the resolution of $1 \times 1 \times 1$ mm^3. Thus, both T1w and T2w scans in the training dataset have volumetric size of $144 \times 192 \times 256$. The scans were already preprocessed for skull stripping, intensity inhomogeneity correction, and cerebellum and brain stem removal before segmenting them manually. We use this dataset for multi-modality volumetric brain tissue segmentation.

Figure 4.3 presents the overlapping intensity distribution of different labels of brain tissue in the training set of iSeg-2017 dataset. Segmentation of brain tissue from MR images specifically in infants is considered challenging due to the presence of highly overlapping intensities between the tissues, lower signal-to-noise ratio, and increased partial volume effects because of relatively large voxel size with respect to the brain size. When the voxel size is large, a voxel may contain multiple types of tissues, thus posing a difficulty in classification or segmentation of the voxel. This effect is known as partial volume effect.



(a)



(b)

Fig. 4.3 Intensity distribution of (a) T1-weighted MR scans (b) T2-weighted MR scans of iSeg-2017 training dataset.

4.2 Preprocessing

CNNs are capable of abstracting latent features from the input data without the need for extensive preprocessing. However, applying transformations on raw data can speed up training and lead to better optimization. The MR scans from both the datasets are already skull-stripped, thus contain only the brain tissues and the background. As preprocessing, we standardize each modality of individual subject as follows.

$$\hat{T} = \frac{T - mean(T_t)}{std(T_t)} \tag{4.1}$$

where $\hat{T}$ is the standardized T1w or T2w scan of a subject, T_t is the region containing brain tissue only, $mean(\cdot)$ and $std(\cdot)$ are the mean and the standard deviation of the input, respectively. Standardization rescales the data value by zero centering it with unit standard deviation. It ensures the data has similar distribution. In case of gradient descent optimization, standardization gives similar weightage to all the features in the data leading to similar update rate for all the parameters, thus avoiding overshoots and improving convergence.

4.3 Weight Initialization

Weight initialization is an important aspect of training a convolutional neural network. Initializing smaller weights for the parameters of the network leads to diminishing outputs after every pass through a layer and the problem gets severe in deep networks where the feature values at the later layers become too small to be significant. This also might lead the network to vanishing gradient problem during backpropagation where the gradient becomes too small that prevents the parameters from updating. In the worst case, the network will cease to learn. Similarly, for larger weights initialization, the output keeps getting larger at the later layers making the network unstable and very sensitive to the changes in input, and subjecting the network to the saturation problem especially if the network has certain activation functions such as sigmoid or tanh. This also might subject the network to the exploding gradient problem where the large gradients will result in large updates to the

network parameters and make the network unstable. In the extreme case, the values can be too large as to overflow and become unrepresentable or undefined. Xavier et al. [84] proposed an initialization method to keep the variance of input and output of a layer similar thus alleviating the problem of the diminishing or increasing outputs at later layers in CNNs. This method samples weight from a normal distribution with standard deviation equal in value to the inverse of the average of the number of input and output connections. As a result, the layers having a smaller number of weights have larger parameter values and vice-versa. He et al. extended the idea for the networks that use non-linear activation functions such as ReLU [85]. This initialization approach is like the Xavier et al., however, it considers the standard deviation equal to two times the inverse of the number of input connections as the ReLU activation is zero for half of the input. We use He et al. method to initialize the parameters of the proposed network.

4.4 Training

We train the proposed network architecture for single-modality and multi-modality segmentation of the brain tissue into three labels—white matter (WM), gray matter (GM), and cerebrospinal fluid (CSF). The input to the network is randomly cropped sub-volumes of size $64 \times 64 \times 64$ from the preprocessed T1w and/or T2w MR scans. We use a mini-batch size of 2 for the training. The small size of input reduces the memory requirements and significantly increases the number of training samples, thus reducing the requirement for data augmentation to generate more data. The network parameters are initialized using He et al. (Section 4.3) initialization method and optimized using adam optimizer (Section 2.3.3). We use the first and second order moments, β_1 and β_2, of 0.97 and 0.999, respectively for the adam optimizer and the learning rate, λ, of 2e-4 initially for the training. To fine-tune the parameters, we reduce the learning rate by a factor of 10 after 5000 epochs. To reduce overfitting and keep the network parameters smaller and the network model simpler, we also use L2 regularization while training the network. L2 regularization adds the squared magnitude of the parameters multiplied by a decay parameter to the loss function. For the experiments,

we use the decay parameter of 6e-4 for the L2 normalization. We study the performance of the proposed network model on three types of loss functions—cross-entropy (Section 2.3.1), dice similarity (Section 2.3.1) and a combination of the two. Although cross-entropy loss facilitates easier gradient calculation, it measures the error between the ground-truth and the predicted score at voxel level and sums up the error. This causes it to be biased when the input data is imbalanced among the labels. The dice score measures the relative overlap between the ground-truth and the prediction ranging between 0 to 1 irrespective of the label size, thus can deal with the data imbalance among the labels [86]. Therefore, to account for accurate voxel-level prediction as well as the data imbalance problem, we consider a combination of the cross-entropy and dice similarity losses like [87] to train our network. The combined loss function is given as follows.

$$L_{CE+DSC}(Y,\hat{Y}) = -\frac{1}{N}\sum_{b=1}^{N}(\frac{1}{2}\sum_{c=1}^{M} Y_{b,c}\cdot log(\hat{Y}_{b,c}) + \sum_{c=1}^{M}\frac{2\cdot Y_{b,c}\cdot\hat{Y}_{b,c}}{Y_{b,c}+\hat{Y}_{b,c}}) \tag{4.2}$$

where, M and N represent the total numbers of labels of segmentation and mini-batch size, respectively, $Y_{b,c}$ is the flattened ground truth and has a value of 1 if the patch b belongs to the label c and 0 otherwise, and $\hat{Y}_{b,c}$ is the flattened predicted probability of the patch b belonging to the label c and has a value between 0 and 1.

The proposed network architecture is implemented using the PyTorch package [88] and the codes with trained models are publicly available at https://github.com/basnetr/U-DenseResNet. We trained the network model on a computer with an Intel Core i5-9600K CPU @ 3.70GHz, 16GB of RAM and an NVIDIA RTX 2070 GPU (8GB) for 6000 epochs. The approximate time to train the network for 6000 epochs was 11 hours. The selection of training and testing data is done such that the performance of the proposed network can be compared with that of the existing architectures that use the same datasets for brain tissue segmentation which is discussed below.

Single-modality segmentation

For single-modality segmentation, we train and test our network architecture on IBSR18 dataset that contains T1w brain MR scans. We select the testing set same as the existing architectures, namely, VoxResNet (Section 2.4.1), 3D-like FCN (Section 2.4.1), 3D U-Net (Section 2.4.2), U-SegNet (Section 2.4.2), Modified U-Net (Section 2.4.2) and RP-Net (Section 2.4.2). The subjects numbered 6, 7, 8, 9, 15, 16, 17 and 18 are selected for testing and the rest of the subjects for training.

Multi-modality segmentation

For multi-modality brain tissue segmentation, we train and test the proposed network on iSeg-2017 dataset containing T1w and T2w brain MR scans. Same as the 3D U-Net (Section 2.4.2), DenseVoxNet (Section 2.4.3), and 3D-SkipDenseSeg (Section 2.4.1) architectures, we test the network on subject 9 of the dataset and compare the performance with that of the others. We train the network on the rest of the subjects of the dataset. We also submit our segmentation results to the iSeg organizers for the evaluation and to compare the results with that of published top-ranking architectures on volumetric infant brain tissue segmentation—3D-SkipDenseSeg (Section 2.4.1), HyperDenseNet (Section 2.4.3), and 3D FC-DenseNet (Section 2.4.2).

4.5 Evaluation Metrics

The performance of the proposed architecture is evaluated and compared to that of the existing deep-learning architectures for brain tissue segmentation using the metrics, dice similarity coefficient (DSC), modified Hausdorff distance (MHD) and average surface distance (ASD).

4.5.1 Dice similarity coefficient

Dice similarity coefficient (DSC) [50] is one of the most popularly used metrics in evaluating the performance of segmentation. It is computed as the ratio of twice the number of pixels in

the overlapping region between the predicted segmentation and the ground truth to the total number of pixels in both the predicted segmentation and the ground truth. It is given as

$$DSC(P,G) = \frac{2|P \cap G|}{|P| + |G|} \tag{4.3}$$

where P is the region of predicted segmentation, G is the ground truth region of the segmentation, and $|\cdot|$ represents the number of pixels in the region. The value of DSC ranges from 0 to 1. Higher the DSC score, more accurate the match between the two regions.

4.5.2 Modified Hausdorff distance

Hausdorff distance [89] is the maximum Euclidean distance between two regions. The one-sided Hausdorff distance from a region, A to another region, B is defined as

$$h(A,B) = max_{a \in A} min_{b \in B} \|a - b\| \tag{4.4}$$

where a represents a pixel in A and b in B, and $\|\cdot\|$ represents the Euclidean distance. The bidirectional Hausdorff distance between the two regions A and B is defined as

$$H(A,B) = max(h(A,B), h(B,A)) \tag{4.5}$$

The bidirectional Hausdorff can be sensitive to outliers as it accounts for the maximum distance only. Thus, to make it robust to the outliers, modified Hausdorff distance is defined that uses 95^{th} percentile of one-sided Hausdorff distance for calculation as

$$MHD(A,B) = max(h_{95}(A,B), h_{95}(B,A)) \tag{4.6}$$

where h_{95} denotes the 95^{th} percentile of the Hausdorff distance. Lower the value of MHD, better the match between the two regions.

4.5.3 Average surface distance

The average surface distance (ASD) [53] between two regions is measured as the mean distance from points on a surface of one region to that on the surface of the another.

Let, the distance between a point a to the surface S_B of a region B be defined as

$$d(a,B) = min_{b \in S_B} \|a - b\| \tag{4.7}$$

Then, the ASD for two regions A and B is defined as

$$ASD(A,B) = \frac{1}{2}(\frac{1}{n_A} \sum_{a \in S_A} d(a,B) + \frac{1}{n_B} \sum_{b \in S_B} d(b,A)) \tag{4.8}$$

where S_A and S_B are the surfaces of two regions A and B, respectively, and n_A and n_B are the total number of points in the surface of A and B, respectively. Lower the value of ASD, better the match between the two regions.

4.6 Experimental results and performance comparison

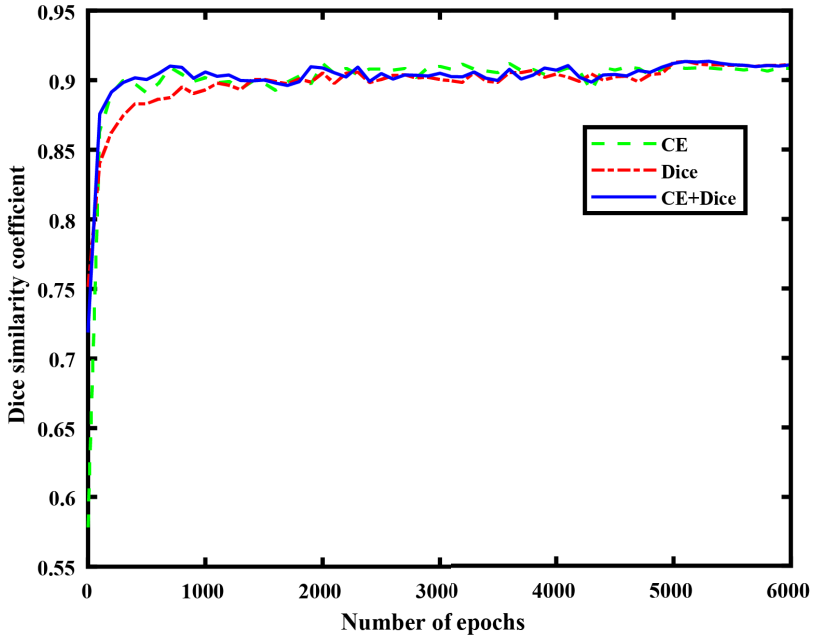
The proposed network is trained using randomly cropped sub-volumes of size $64 \times 64 \times 64$. For IBSR18 dataset, T1w scans are used for single-modality brain tissue segmentation and for iSeg-2017 dataset, both T1w and T2w scans are used for multi-modality brain tissue segmentation. The network is trained independently using the three loss functions, cross-entropy loss (CE), dice-similarity loss (Dice) and a combination of the two (CE+Dice). For testing, the T1w and/or T2w are cropped to the sub-volumes of size $64 \times 64 \times 64$ in the steps of 16 along each dimension of the scans and passed through the network. The outputs of the sub-volumes are combined and the probability scores at the overlapping regions are averaged to obtain the final segmentation. The testing is done on the same test sets as used in the other methods for equitable comparison of the results. The average test time on Intel Core i5-9600K CPU @ 3.70GHz, 16GB of RAM and an NVIDIA RTX 2070 GPU (8GB) for a

forward pass through the proposed network is 50.70 milliseconds per sub-volume and is 36 seconds per subject of size $144 \times 192 \times 256$.

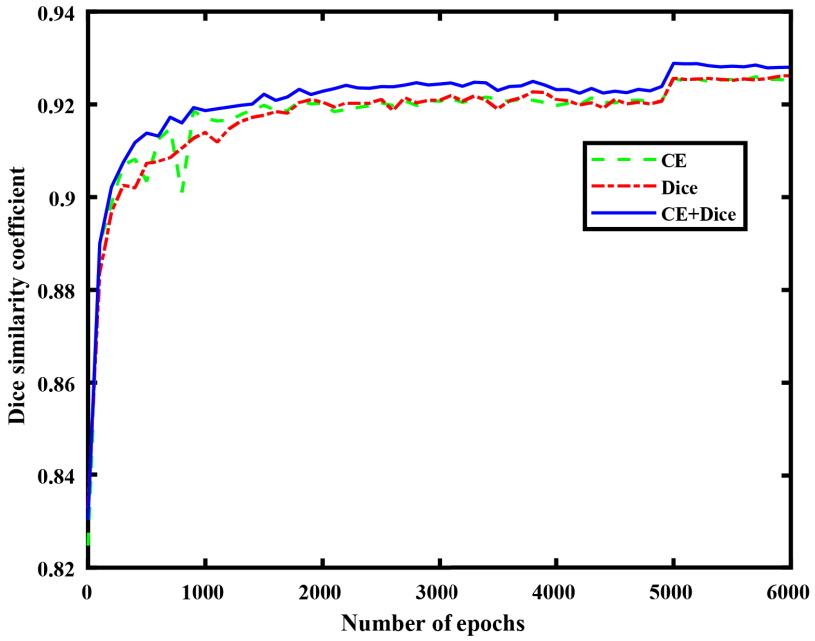
Figures 4.4(a) and 4.4(b) show the segmentation performance of the network in terms of dice similarity coefficient on validation data during the training using three different loss function. It is seen from the figure that the combined loss function converges relatively faster compared to the other two loss functions.

4.6.1 IBSR

Table 4.2 presents the segmentation performance of the proposed method using the three loss functions, CE, Dice and CE+Dice, and a comparison with that of the state-of-the-art deep learning architectures for the single-modality brain tissue segmentation on IBSR18 dataset in terms of dice similarity coefficient (DSC) and the number of parameters. The obtained scores for each label are the average DSC scores of 8 subjects numbered 6, 7, 8, 9, 15, 16, 17, and 18. It is observed from the table that the proposed method achieves a performance that is superior to that of all the existing deep-learning methods in terms of DSC for all the brain tissue labels (WM, GM, and CSF) independently; moreover, using the least number of parameters and irrespective of the type of loss function used during the training. The proposed architecture shows an improvement of about 3% in the average DSC score using the combined loss function as compared to that of the RP-Net, the architecture with the current best performance. The reduction in the number of parameters in the proposed architecture is about 98%, 96%, 95%, 80% and 47% as compared to that of the modified U-Net [75], 3D-UNet [46], 3D-like FCN [42], U-SegNet [77], and VoxResNet [49], respectively. In addition, the average DSC improvement of the proposed architecture using the combined loss function is about 3.8%, 5.1%, 4.9%, 13.5% and 8.4% for the corresponding architectures. From this table, it is also seen that the proposed network trained on the combined loss yields better accuracy, i.e. best DSC score in 2 out of 3 tissue labels and an overall best average DSC score. Figure 4.5 shows the qualitative comparison of the segmentation results on one of the IBSR18 test data, subject 9, using the proposed architecture trained on the three different loss functions. Figure 4.5(a) presents slices of a T1w scan along the three dimensions,



(a)



(b)

Fig. 4.4 Learning curves of the proposed method using three loss functions, the cross-entropy loss (CE), dice similarity loss (Dice), and a combination of cross-entropy and dice similarity loss (CE+Dice), on the (a) IBSR18 validation data and (b) iSeg-2017 validation data.

Table 4.2 Comparison of segmentation performance of the proposed architecture in terms of dice similarity coefficient (DSC) score for three labels of segmentation, WM, GM, and CSF, and their average and the number of parameters on the test set comprising subjects 6 to 9 and 15 to 18 of the IBSR18 dataset. The bold-faced values in red font indicate the results from the best performing methods, and that in blue and green fonts indicate the results from the second-best and third-best performing methods, respectively.

Architecture	DSC				Number of
	WM	GM	CSF	Average	Parameters
VoxResNet [49]	0.8635	0.8817	0.8333	0.8595	**1.33M**
3D-like FCN [42]	0.9004	0.9075	0.8561	0.8880	15M
3D-UNet [46]	0.8997	0.9095	0.8499	0.8864	19M
U-SegNet [77]	0.8923	0.9033	0.6658	0.8205	**3.48M**
Modified U-Net [75]	0.9070	0.9185	0.8678	0.8978	31.03M
RP-Net [50]	0.9121	0.9208	0.8818	0.9049	-
Proposed (CE)	**0.9399**	**0.9447**	**0.8960**	**0.9269**	**0.71M**
Proposed (Dice)	**0.9367**	**0.9394**	**0.9048**	**0.9270**	**0.71M**
Proposed (CE+Dice)	**0.9388**	**0.9452**	**0.9105**	**0.9315**	**0.71M**

Figure 4.5(b) presents the ground truth of the segmentation, and Figures 4.5(c), 4.5(d), and 4.5(e) present the segmentation results for the proposed network using the three loss functions. It is seen from the figures that good segmentation results can be obtained using the proposed network irrespective of the loss function used in training, and the combined loss function yields results closer to the ground truth than that using the other two loss functions as seen more clearly from the circled regions of the figures.

4.6.2 iSeg

Table 4.3 presents the segmentation performance of the proposed method using the three loss functions, CE, Dice and CE+Dice, and a comparison with that of the state-of-the-art deep learning architectures for the multi-modality brain tissue segmentation on iSeg-2017 dataset in terms of dice similarity coefficient (DSC), depth of the network and the number of parameters. The obtained results for each label are the DSC scores of the test subject 9. It is observed from the table that the proposed method achieves performance that is superior to that of all the existing deep-learning methods in terms of DSC for all the brain tissue labels

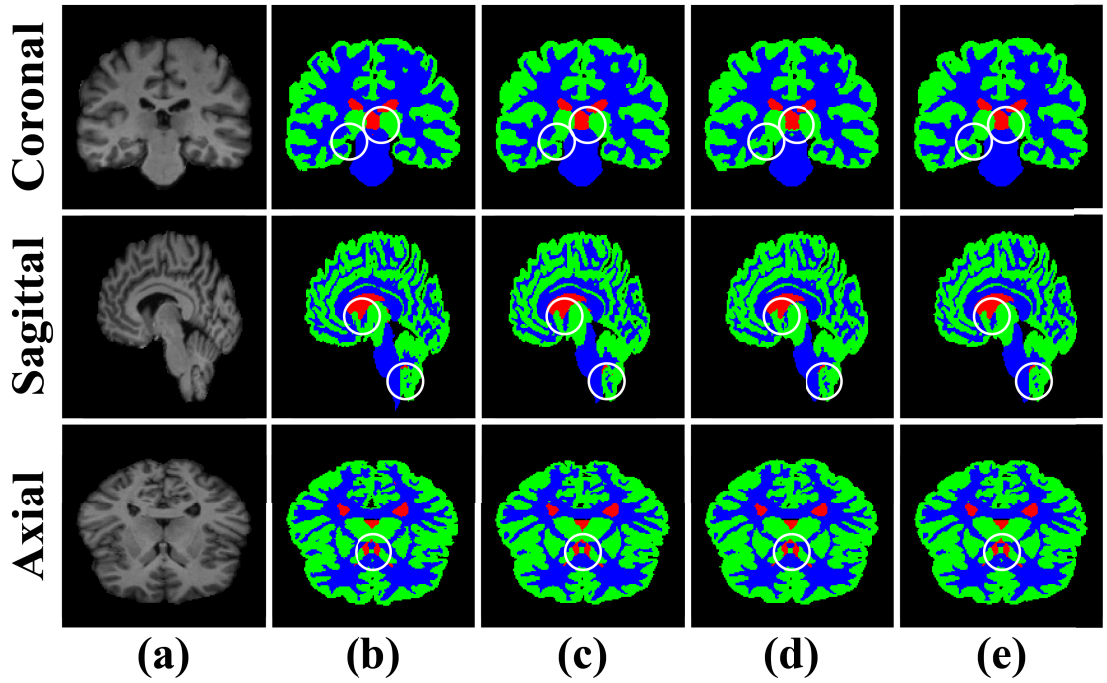


Fig. 4.5 Qualitative illustration of the proposed segmentation scheme on the subject 9 test set of the IBSR18 dataset. (a) T1 images. (b) Ground truth images. c) Images obtained using the cross-entropy loss. (d) Images obtained using the dice similarity loss. (e) Images obtained using a combination of the cross-entropy and dice similarity losses.

(WM, GM, and CSF) independently; moreover, using the least number of parameters and irrespective of the type of loss function used during the training. The proposed architecture shows an improvement of about 0.76% in the average DSC score using the combined loss function as compared to that of the 3D FC-DenseNet, the architecture with the current best performance. Even compared to the method using the least number of parameters at present, VoxResNet, the reduction in the number of parameters in the proposed architecture is about 47% with an increase in the average DSC score by about 1.7%. The reduction in the number of parameters in the proposed architecture is about 97%, 85% and 54% as compared to that of the 3D-UNet [46], DenseVoxNet [52], and 3D-SkipDenseSeg [53], respectively, with some improvement in the DSC score. From this table, it can also be seen that the proposed network trained on the combined loss yields better accuracy, i.e. best DSC score in 2 out of 3 tissue labels and an overall best average DSC score. Figure 4.6 shows the qualitative comparison of the segmentation results on one of the iSeg2017 test data, subject 9, using the proposed architecture trained on the three different loss functions. Figure 4.6(a) presents slices of a T1w scan along the three dimensions, Figure 4.6(b) presents the ground truth of the segmentation, and Figures 4.6(c), (d) and (e) present the segmentation results for the proposed network using the three loss functions. It is seen from the figures that good segmentation results can be obtained using the proposed network irrespective of the loss function used in training, and the combined loss function yields results closer to the ground truth than that using the other two loss functions as seen more clearly from the circled regions of the figures.

We also participated in the MICCAI Grand Challenge on 6-month infant brain MRI Segmentation[4] and sent our segmentation results on the organizers' test subjects to evaluate our performance as the manual segmentation of these subjects are not publicly released. The organizers of the challenge report the performance scores in terms of dice similarity coefficient (DSC), Modified Hausdorff Distance (MHD), and Average Surface Distance (ASD).

[4]http://iseg2017.web.unc.edu/rules/results/

Table 4.3 Comparison of segmentation performance of the proposed architecture in terms of dice similarity coefficient (DSC) score for three labels of segmentation, WM, GM, and CSF, and their average, depth of the network, and the number of parameters on the test set (subject 9) of iSeg-2017 dataset. The bold-faced values in red font indicate the results from the best performing methods, and that in blue and green fonts indicate the results from the second-best and third-best performing methods, respectively.

Architecture	DSC				Depth	Number of Parameters
	WM	GM	CSF	Average		
3D-UNet [46]	0.8957	0.9073	0.9444	0.9158	18	19M
VoxResNet [49]	0.8987	0.9064	0.9428	0.9160	25	**1.33M**
DenseVoxNet [52]	0.8546	0.8851	0.9371	0.8923	32	4.34M
3D-SkipDenseSeg [53]	0.9101	0.9164	0.9488	0.9218	47	1.55M
3D FC-DenseNet [55]	0.9037	0.9179	0.9519	0.9245	60	**1.4M**
Proposed (CE)	**0.9165**	**0.9221**	**0.9527**	**0.9305**	52	**0.71M**
Proposed (Dice)	**0.9174**	**0.9199**	**0.9495**	**0.9289**	52	**0.71M**
Proposed (CE+Dice)	**0.9159**	**0.9228**	**0.9558**	**0.9315**	52	**0.71M**

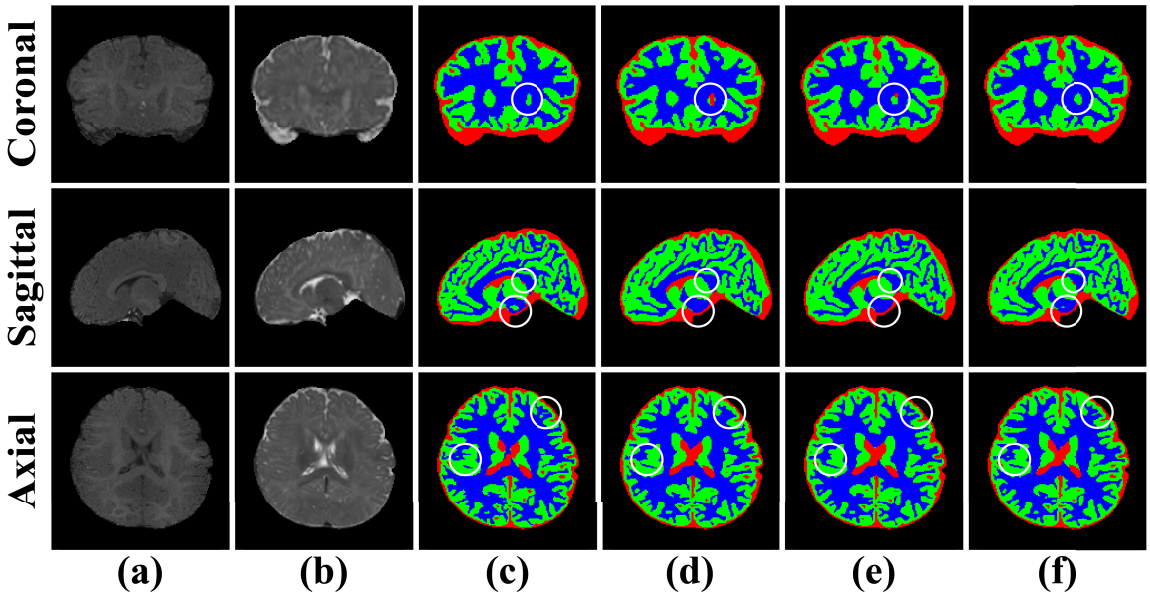


Fig. 4.6 Qualitative illustration of the proposed segmentation scheme on the subject 9 test set of the iSeg-2017 dataset. (a) T1 images. (b) T2 images. (b) Ground truth images. c) Images obtained using the cross-entropy loss. (d) Images obtained using the dice similarity loss. (e) Images obtained using a combination of the cross-entropy and dice similarity losses.

Table 4.4 Comparison of segmentation performance of proposed architecture with published deep learning architectures in terms of metrics dice similarity coefficient (DSC), modified Hausdorff distance (MHD), and average surface distance (ASD) and the number of parameters on iSeg-2017 test dataset. The bold-faced values in red font indicate the results from the best performing methods, and that in blue and green fonts indicate the results from the second-best and third-best performing methods, respectively.

Method	WM			GM			CSF			Number of
	DSC	MHD	ASD	DSC	MHD	ASD	DSC	MHD	ASD	Parameters
MSL_SKKU [53]	**0.904**	**6.618**	**0.375**	**0.923**	**5.999**	**0.321**	**0.958**	**9.112**	**0.116**	**1.55M**
BCH_CLR_IMAGINE [55]	**0.907**	7.104	**0.360**	**0.926**	9.557	**0.311**	**0.960**	**8.850**	**0.110**	**1.4M**
HyperDenseNet [54]	0.901	**6.660**	**0.382**	**0.920**	**5.752**	**0.329**	0.956	9.421	0.120	10M
Proposed(CE+Dice)(Max-Pool)	**0.903**	**6.778**	0.399	0.919	**5.694**	**0.329**	**0.957**	**9.201**	**0.119**	**0.63M**

Table 4.5 Comparison of segmentation performance of our architecture in terms of dice similarity coefficient (DSC) score using different downsampling layers on test data of iSeg-2017 dataset (subject 9). The bold-faced values in red font indicate the results from the best performing methods.

Proposed Architecture	DSC				Depth	Number of
	WM	GM	CSF	Average		Parameters
CE+Dice (Max-Pool)	**0.9192**	0.9217	0.9519	0.9309	48	**0.63M**
CE+Dice (Conv-Pool)	0.9159	**0.9228**	**0.9558**	**0.9315**	52	0.71M

Table 4.4 shows the average performance scores of the proposed network on the all the test subjects of the iSeg-2017 challenge and a comparison to that of the the top-ranking published deep learning methods. It should be noted that during the time of submission of our results to the organizers of the segmentation challenge, we used max-pooling layers instead of convolution layers for downsampling, thus network architecture had 625,926 parameters only with 48 layers [90]. The network was trained on the combined loss function. Due to the limitation on the number of submissions allowed for the competition, we were not able to submit the results of the proposed architecture that uses convolutional layers as downsamplers. It is observed from the table that our method that used max-pooling for downsampling has the least number of parameters that is about 59%, 55% and 94% less than the number of parameters in [53], [55],and [54], respectively with a decrease in the average accuracy in terms of the DSC score by only 0.2%, 0.5% and 0%, respectively, and in terms of ASD by 4.1%, 8.5% and 1.8%, respectively. Nevertheless, there is an increase in the average accuracy in terms of MHD by 0.3%, 15.1%, and 0.7% compared to that of the corresponding methods. Table 4.5 shows a comparison of the performance of our architecture that uses the max-pooling layer with that using the convolutional layer for downsampling on the test data (subject 9) of the iSeg-2017 dataset. It observed from this table, as also mentioned in [47], that using the convolutional layer for downsampling instead of the max-pooling layer enhances the performance in deeper convolutional neural networks with a slight increase in the number of parameters.

4.7 Summary

In this chapter, the experimental setup for testing the segmentation performance of the proposed architecture on two different datasets containing MR scans of the brain tissues of diverse age groups has been discussed. The experiments are performed and results of the experiments including the segmentation performance of the proposed architecture and a comparison with that of the existing methods of the brain tissue segmentation have been presented in terms of the number of parameters and the evaluation metrics that are

commonly used to measure performance of segmentation. It has been shown that the proposed method outperforms all the state-of-the-art deep learning architectures used to segment brain tissues in terms of the evaluation metric for all the brain tissue labels (WM, GM, and CSF) independently using the least number of parameters for the test sets of both the IBSR18 and iSeg-2017 datasets. It has also been shown that the combined loss serves as the best loss function for training the network. In the iSeg-2017 challenge, the proposed method has shown overall a competitive performance with the least number of parameters and the best performance in terms of one of the evaluation metrics among all the deep learning based architectures in the literature.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this thesis, a novel three-dimensional parameter-efficient deep convolutional neural network based on dense and residual connections has been proposed for volumetric brain tissue segmentation from MR images. Structural information obtained from the segmentation of the brain tissues from magnetic resonance imaging (MRI) facilitates clinical research and diagnosis not limited to disease identification, anatomical and functional study, abnormality and disorder detection, and image-guided surgical interventions. MR image modalities such as T1-weighted and T2-weighted scans provide various contrast information that can be effectively utilized for tissue segmentation. However, segmentation of the brain tissues from the MR images is challenging since the imaging modality suffers from intensity inhomogeneities and overlaps, low signal-to-noise ratio, and artifacts like partial volume effects. Recently, deep-learning based methods, especially the algorithms employing convolutional neural network architectures (CNNs), have been actively used for the segmentation of the brain tissues from MR images. CNNs are capable of abstracting complex representations at various levels that is well suited for image processing problems such as segmentation. Different convolutional layers with various kernel sizes, pooling layers, and transpose convolutional layers can capture finer local details to semantic contextual information from the input data. CNNs used in the medical image processing including brain tissue segmentation

have been innovated over the course of time to increase the representation capacity of the network. For example, to capture the rich spatial information present in the volumetric scans of different imaging modalities, 2D CNNs have been replaced by 3D CNNs. Similarly, the numbers of layers in the CNNs have been increased for improved representation and kernel size decreased to allow the design of deeper networks. Architectures like U-Net introduced pooling and corresponding upsampling layers that have been placed symmetrically in between convolutional layers to effectively capture latent features at various scales, and the skip-connections to allow effective information transfer from earlier part of the network containing high-resolution local information to the latter part for segmentation. Furthermore, the typically stacked convolutional layers have been replaced by convolutional layers with dense and residual connections to improve information and gradient flow in the network while addressing the issues such as degradation problem and vanishing or exploding gradient problems. While attempting to the increase representation capacity of the network, there is also a need to account for the computational expense and complex optimization required to train and test the network. Factors such as number of layers, size of kernels in the layers, and the connection between the layers that impact the number of parameters in the CNNs can be efficiently selected or designed to make the network more robust. The proposed network architecture has used dense and residual connections at various parts of the network with step-wise contextualization and localization of abstracted features, and skip-connections to facilitate effective information flow between the layers of the network for increased representation and efficient parameter reduction. It has utilized small-sized kernels in a deep CNN architecture with dense and residual connections reducing the number of parameters significantly for volumetric segmentation of brain tissues into white matter, gray matter, and cerebrospinal fluid. A sub-volume or patch-based approach has been used during training and testing of the proposed network to reduce the memory requirement. Furthermore, the network has been trained on three different loss functions, cross-entropy, dice similarity, and a combination of the two, independently to find the best loss function for optimization. The network architecture has been evaluated for single-modality and multi-modality brain tissue segmentation using the datasets IBSR18 and iSeg-2017, respectively. Experimental results

on the test set of both datasets have shown that the combined loss function yields superior performance in terms of popular evaluation metrics used in segmentation. A comparison of performance with the state-of-the-art methods in the test set of both datasets have shown that the proposed method provides the best performance. In the MICCAI Grand Challenge for 6-month infant brain MRI Segmentation, the proposed method has achieved competitive results compared to the top-ranking published architectures while outperforming them in some metrics. The proposed architecture has reduced the number of parameters significantly, ranging from 47% to 98%, compared to the popularly used deep learning architectures in single- and multi-modality volumetric brain tissue segmentation from MR images. In conclusion, the proposed method is automatic, provides reliable segmentation accuracy, is highly parameter-efficient, and can work in multi-modal settings. Thus, the proposed method can be expected to play a significant role in real-time clinical diagnosis and research.

5.2 Future Work

This research has several possibilities of improvement for further study in the future. In this thesis, the proposed network architecture was utilized for segmenting three labels of tissues in the human brain from the MRI scans. The network can be extended to segment the tissues into multiple labels such as various anatomical regions of the human body at once. We have used three different loss functions, namely, cross-entropy loss, dice similarity loss and a combination of the two for optimizing the network architecture. In the future, carefully defined loss functions such as weighted loss functions can be used to train the network model to address the data imbalance problem among the labels. Similarly, the network model can be made more robust by employing deep supervision [74] for training.

References

[1] M. Joliot and B. M. Mazoyer, “Three-dimensional segmentation and interpolation of magnetic resonance brain images," *IEEE Trans. Med. Imag,* vol. 12, no. 2, pp. 269-277, Jun. 1993, DOI: 10.1109/42.232255.

[2] L. Dora, S. Agrawal, R. Panda, and A. Abraham, “State-of-the-art methods for brain tissue segmentation: A review," *IEEE Rev. Biomed. Eng.,* vol. 10, pp. 235-249, Jun. 2017, DOI: 10.1109/RBME.2017.2715350.

[3] C. Fennema-Notestine *et al.,* “Structural MRI biomarkers for preclinical and mild Alzheimer’s disease," *Human Brain Mapping,* vol. 30, no. 10, pp. 3238-3253, Mar. 2009, DOI: 10.1002/hbm.20744.

[4] J. Tanabe *et al.,* “Tissue segmentation of the brain in Alzeihmer’s disease," *J. Neuroradiol.,* vol. 18, no. 1, pp. 115-123, Jan. 1997.

[5] F. G. Woermann, S. L. Free, M. J. Koepp, S. M. Sisodiya, and J. S. Duncan, “Abnormal cerebral structure in juvenile myoclonic epilepsy demonstrated with voxel-based analysis of MRI,” *Brain,* vol. 122, no. 11, pp. 2101-2108, Nov. 1999, DOI: 10.1093/brain/122.11.2101.

[6] M. E. Shenton *et al.,* “Abnormalities of the left temporal lobe and thought disorder in schizophrenia: a quantitative magnetic resonance imaging study, " *New England Journal of Medicine,* vol. 327, no. 9, pp. 604-612, Aug. 1992, DOI: 10.1056/NEJM199208273270905.

[7] A. P. Zijdenbos, B. M. Dawant, R. A. Margolin, and A. C. Palmer, “Morphometric analysis of white matter lesions in MR images: method and validation," *IEEE Trans. Med. Imag.,* vol. 13, no. 4, pp. 716-724, Dec. 1994, DOI: 10.1109/42.363096.

[8] M. C. Keuken, P.-L. Bazin, A. Schäfer, J. Neumann, R. Turner, and B. U. Forstmann, “Ultra-high 7T MRI of structural age-related changes of the subthalamic nucleus," *Journal of Neuroscience,* vol. 33, no. 11, pp. 4896-4900, Mar. 2013, DOI: 10.1523/JNEUROSCI.3241-12.2013.

[9] R. Goebel, F. Esposito, and E. Formisano, “Analysis of functional image analysis contest (FIAC) data with brainvoyager qx: From single-subject to cortically aligned group general linear model analysis and self-organizing group independent component analysis," *Human Brain Mapping,* vol. 27, no. 5, pp. 392-401, May 2006, DOI: 10.1002/hbm.20249.

[10] A. Makropoulos, S. J. Counsell, and D. Rueckert, “A review on automatic fetal and neonatal brain MRI segmentation,” *NeuroImage,* vol. 170, pp. 231-248, Apr. 2018, DOI: 10.1016/j.neuroimage.2017.06.074.

[11] P. Kalavathi, “Brain tissue segmentation in MR brain images using multiple Otsu’s thresholding technique," in *Int. Conf. Comput. Sci. Edu.,* Colombo, Sri Lanka, 2013, pp. 639-642.

[12] A. Wadhwa, A. Bhardwaj, and V. Singh Verma, “A review on brain tumor segmentation of MRI images,” *Magnetic Resonance Imaging,* vol. 61, pp. 247-259, Sept. 2019, DOI: 10.1016/j.mri.2019.05.043.

[13] I. Despotović, B. Goossens, and W. Philips, “MRI segmentation of the human brain: challenges, methods, and applications,” *Comput. Math. Methods Med.,* vol. 2015, pp. 450341-450341, Mar. 2015, DOI: 10.1155/2015/450341.

[14] R. Hiralal and H. P. Menon, “A survey of brain MRI image segmentation methods and the issues involved,” in *Int. Symp. Intell. Syst. Technol. Appl.,* Jaipur, India, 2016, pp. 245-259.

[15] M. Rouaïnia, M. S. Medjram, and N. Doghmane, “Brain MRI Segmentation and lesions detection by EM algorithm,” *Int. J. Med. Health Sci.,* vol. 2, no. 24, pp. 379-382, Dec. 2008, DOI: 10.5281/zenodo.1328952.

[16] U.-S. Choi, H. Kawaguchi, Y. Matsuoka, T. Kober, and I. Kida, “Brain tissue segmentation based on MP2RAGE multi-contrast images in 7 T MRI," *PloS one,* vol. 14, no. 2, pp. e0210803, Feb. 2019, DOI: 10.1371/journal.pone.0210803.

[17] Z. Pan and J. Lu, “A Bayes-based region-growing algorithm for medical image segmentation," *Comput. Sci. Eng.,* vol. 9, no. 4, pp. 32-38, Jul. 2007, DOI: 10.1109/MCSE.2007.67.

[18] J. Yang and S. Huang, “Method for evaluation of different MRI segmentation approaches," *IEEE Transactions on Nuclear Science,* vol. 46, no. 6, pp. 2259-2265, Dec. 1999, DOI: 10.1109/23.819313.

[19] H. Greenspan, A. Ruf, and J. Goldberger, “Constrained Gaussian mixture model framework for automatic segmentation of MR brain images,” *IEEE Trans. Med. Imag.,* vol. 25, no. 9, pp. 1233-1245, Sept. 2006, DOI: 10.1109/TMI.2006.880668.

[20] L. Wang, F. Shi, W. Lin, J. H. Gilmore, and D. Shen, “Automatic segmentation of neonatal images using convex optimization and coupled level sets," *Neuroimage,* vol. 58, no. 3, pp. 805-817, Oct. 2011, DOI: 10.1016/j.neuroimage.2011.06.064.

[21] F. Shi, Y. Fan, S. Tang, J. H. Gilmore, W. Lin, and D. Shen, “Neonatal brain image segmentation in longitudinal MRI studies," *NeuroImage,* vol. 49, no. 1, pp. 391-400, Jan. 2010, DOI: 10.1016/j.neuroimage.2009.07.066.

[22] P. Moeskops *et al.,* “Automatic segmentation of MR brain images of preterm infants using supervised classification," *NeuroImage,* vol. 118, pp. 628-641, Sept. 2015, DOI: 10.1016/j.neuroimage.2015.06.007.

[23] M. De Craene, A. du Bois d'Aische, B. Macq, and S. K. Warfield, "Multi-subject registration for unbiased statistical atlas construction," in *Proc. Med. Image Comput. Comput.-Assist. Intervent.,* Saint-Malo, France, 2004, pp. 655-662.

[24] L. Sun, L. Zhang, and D. Q. Zhang, "Multi-atlas based methods in brain MR image segmentation," *Chin. Med. Sci. J.,* vol. 34, no. 2, pp. 110-119, Jun. 2019, DOI: 10.24920/003576.

[25] M. Cabezas, A. Oliver, X. Lladó, J. Freixenet, and M. Bach Cuadra, "A review of atlas-based segmentation for magnetic resonance brain images," *Computer Methods and Programs in Biomedicine,* vol. 104, no. 3, pp. e158-e177, Dec. 2011, DOI: 10.1016/j.cmpb.2011.07.015.

[26] J. Morin, C. Desrosiers, and L. Duong, "Atlas-based segmentation of brain magnetic resonance imaging using random walks," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.,* Providence, RI, USA, 2012, pp. 44-49.

[27] Y. Le Cun *et al.,* "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541-551, Dec. 1989, DOI: 10.1162/neco.1989.1.4.541.

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton. (2012). ImageNet classification with deep convolutional neural networks. presented at the Neural Information Processing Systems 2012. [Online]. Available: https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks

[29] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.,* Venice, Italy, 2017, pp. 2980-2988.

[30] C. Ledig *et al.,* "Photo-realistic single image super-resolution using a generative adversarial network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.,* Honolulu, HI, USA, 2017, pp. 105-114.

[31] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.,* Boston, MA, USA, 2015, pp. 3431-3440.

[32] G. Litjens *et al.,* “A survey on deep learning in medical image analysis," *Med. Image Anal.,* vol. 42, pp. 60-88, Dec. 2017, DOI: 10.1016/j.media.2017.07.005.

[33] M. H. Hesamian, W. Jia, X. He, and P. Kennedy, “Deep learning techniques for medical image segmentation: Achievements and challenges," *J. Digit. Imag.,* vol. 32, no. 4, pp. 582-596, Aug. 2019, DOI: 10.1007/s10278-019-00227-x.

[34] D. Shen, G. Wu, and H.-I. Suk, “Deep learning in medical image analysis," *Annu. Rev. Biomed. Eng.,* vol. 19, no. 1, pp. 221-248, Mar. 2017, DOI: 10.1146/annurev-bioeng-071516-044442.

[35] Y.-A. Chung and W.-H. Weng. (2017). Learning deep representations of medical images using siamese cnns with application to content-based image retrieval. arXiv preprint. [Online]. Available: https://ui.adsabs.harvard.edu/abs/2017arXiv171108490C

[36] W. Zhang *et al.,* “Deep convolutional neural networks for multi-modality isointense infant brain image segmentation,” *NeuroImage,* vol. 108, pp. 214-224, Mar. 2015, DOI: 10.1016/j.neuroimage.2014.12.061.

[37] D. Nie, L. Wang, Y. Gao, and D. Shen, “Fully convolutional networks for multi-modality isointense infant brain image segmentation,” in *Proc. IEEE Int. Symp. Biomed. Imaging,* Prague, Czech Republic, 2016, pp. 1342-1345.

[38] A. Prasoon, K. Petersen, C. Igel, F. Lauze, E. Dam, and M. Nielsen, “Deep feature learning for knee cartilage segmentation using a triplanar convolutional neural network," in *Proc. Med. Image Comput. Comput.-Assist. Intervent.,* Nagoya, Japan, 2013, pp. 246-253.

[39] P. Moeskops, M. A. Viergever, A. M. Mendrik, L. S. d. Vries, M. J. N. L. Benders, and I. Išgum, “Automatic segmentation of MR brain images With a convolutional neural

network," *IEEE Trans. Med. Imag.,* vol. 35, no. 5, pp. 1252-1261, May 2016, DOI: 10.1109/TMI.2016.2548501.

[40] K. Kamnitsas *et al.,* "Efficient multi-scale 3D CNN with fully connected CRF for accurate brain lesion segmentation," *Med. Image Anal.,* vol. 36, pp. 61-78, Feb. 2017, DOI: 10.1016/j.media.2016.10.004.

[41] J. Chen, L. Yang, Y. Zhang, M. Alber, and D. Z. Chen, "Combining fully convolutional and recurrent neural networks for 3D biomedical image segmentation, " in *Proc. Adv. Neural Inf. Process. Syst.,* Barcelona, Spain, 2016, pp. 3036-3044.

[42] Y. Xu, T. Géraud, and I. Bloch, "From neonatal to adult brain MR image segmentation in a few seconds using 3D-like fully convolutional network and transfer learning," in *Proc. ICIP,* Beijing, China, 2017, pp. 4417-4421.

[43] G. Urban, M. Bendszus, F. Hamprecht, and J. Kleesiek, "Multi-modal brain tumor segmentation using deep convolutional neural networks," in *Proc. MICCAI Brain Tumor Segmentation Challenge,* 2014, pp. 31-35.

[44] J. Dolz, C. Desrosiers, and I. Ben Ayed, "3D fully convolutional networks for subcortical segmentation in MRI: A large-scale study," *NeuroImage,* vol. 170, pp. 456-470, Apr. 2018, DOI: 10.1016/j.neuroimage.2017.04.039.

[45] O. Ronneberger, P. Fischer, and T. Brox. (2015). U-Net: Convolutional networks for biomedical image segmentation. presented at Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015. [Online]. Available: https://ui.adsabs.harvard.edu/abs/2015arXiv150504597R

[46] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, "3D U-Net: Learning dense volumetric segmentation from sparse annotation," in *Proc. Med. Image Comput. Comput.-Assist. Intervent.,* Athens, Greece, 2016, pp. 424-432.

[47] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-Net: Fully convolutional neural networks for volumetric medical image segmentation," in *2016 Fourth International Conference on 3D Vision (3DV),* Stanford, CA, USA, 2016, pp. 565-571.

[48] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.,* Las Vegas, NV, USA, 2016, pp. 770-778.

[49] H. Chen, Q. Dou, L. Yu, J. Qin, and P.-A. Heng, "VoxResNet: Deep voxelwise residual networks for brain segmentation from 3D MR images," *NeuroImage,* vol. 170, pp. 446-455, Apr. 2018, DOI: 10.1016/j.neuroimage.2017.04.04.

[50] L. Wang, C. Xie, and N. Zeng, "RP-Net: A 3D convolutional neural network for brain segmentation from magnetic resonance imaging," *IEEE Access,* vol. 7, pp. 39670-39679, Mar. 2019, DOI: 10.1109/ACCESS.2019.2906890.

[51] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.,* Honolulu, HI, USA, 2017, pp. 2261-2269.

[52] L. Yu *et al.,* "Automatic 3D cardiovascular MR segmentation with densely-connected volumetric ConvNets," in *Med. Image Comput. Comput. Assist. Interv,* Quebec City, QC, Canada, 2017, pp. 287-295.

[53] T. D. Bui, J. Shin, and T. Moon, "Skip-connected 3D DenseNet for volumetric infant brain MRI segmentation," *Biomedical Signal Processing and Control,* vol. 54, p. 101613, Sept. 2019, DOI: 10.1016/j.bspc.2019.101613.

[54] J. Dolz, K. Gopinath, J. Yuan, H. Lombaert, C. Desrosiers, and I. B. Ayed, "HyperDense-Net: A hyper-densely connected CNN for multi-modal image segmentation," *IEEE Trans. Med. Imag.,* vol. 38, no. 5, pp. 1116-1126, May 2019, DOI: 10.1109/TMI.2018.2878669.

[55] S. Raein Hashemi, S. P. Prabhu, S. K. Warfield, and A. Gholipour, “Exclusive independent probability estimation using deep 3D fully convolutional DenseNets: Application to isointense infant brain MRI segmentation,” in *Proc. 2nd Int. Conf. Med. Imag. Deep Learn.,* London, England, 2019, pp. 1-14.

[56] W. S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity," *Bull. Math. Biophys.,* vol. 5, no. 4, pp. 115-133, Dec. 1943, DOI: 10.1007/BF02478259.

[57] F. Rosenblatt, “The perceptron: A probabilistic model for information storage and organization in the brain," *Psychol. Rev.,* vol. 65, no.6, pp. 386-408, Nov. 1958, DOI: 10.1037/h0042519.

[58] I. Goodfellow, Y. Bengio, and A. Courville, “Deep learning,” Cambridge, MA, USA:MIT Press, 2016.

[59] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors," *Nature,* vol. 323, no. 6088, pp. 533-536, Oct. 1986, DOI: 10.1038/323533a0.

[60] V. Dumoulin and F. Visin. (2016). A guide to convolution arithmetic for deep learning. arXiv e-prints. [Online]. Available: https://ui.adsabs.harvard.edu/abs/2016arXiv160307285D

[61] V. Nair and G. E. Hinton, “Rectified linear units improve restricted Boltzmann machines," in *Proc. Int. Conf. Mach. Learn.,* Haifa, Israel, 2010, pp. 807-814.

[62] X. Glorot, A. Bordes, and Y. Bengio, “Deep sparse rectifier neural networks," in *Proc. Int. Conf. Artif. Intell. Stat.,* Fort Lauderdale, FL, USA, 2011, pp. 315-323.

[63] Y.-L. Boureau, J. Ponce, and Y. LeCun, “A theoretical analysis of feature pooling in visual recognition," in *Proc. Int. Conf. Mach. Learn.,* Haifa, Israel, 2010, pp. 111-118.

[64] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. (2014). Striving for simplicity: The all convolutional net. arXiv preprint. [Online]. Available: https://ui.adsabs.harvard.edu/abs/2014arXiv1412.6806S

[65] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.,* vol. 15, no. 1, pp. 1929-1958, Jan. 2014, DOI: 10.5555/2627435.2670313.

[66] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods, *USSR Comput. Math. Math. Phys.,* vol. 4, no. 5, pp. 1-17, 1964, DOI: 10.1016/0041-5553(64)90137-5.

[67] Y. E. Nesterov, "A method of solving a convex programming problem with convergence rate O(1/k^2)," *Dokl. akad. nauk Sssr,* vol. 269, no. 3, pp. 543–547, 1983.

[68] I. Sutskever, J. Martens, G. E. Dahl, and G. E. Hinton, "On the importance of initialization and momentum in deep learning," in *Proc. Int. Conf. Mach. Learn. (ICML),* Atlanta, GA, USA, 2013, pp. 1139-1147.

[69] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.,* vol. 12, no. 7, pp. 2121-2159, Jul. 2011, DOI: 10.5555/1953048.2021068.

[70] M. D. Zeiler. (2012). ADADELTA: An adaptive learning rate method. arXiv preprint. [Online]. Available: https://ui.adsabs.harvard.edu/abs/2012arXiv1212.5701Z

[71] G. Hinton, N. Srivastava, and K. Swersky, "Neural networks for machine learning—Lecture 6a: Overview of mini-batch gradient descent, " Toronto, ON, Canada, Jul. 2012, [Online] Available: https://www.cs.toronto.edu/tijmen/csc321/slides/lecture_slides_lec6.pdf

[72] D. P. Kingma and J. Ba. (2014). Adam: A method for stochastic optimization. arXiv preprint. [Online]. Available: https://ui.adsabs.harvard.edu/abs/2014arXiv1412.6980K

[73] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Representations,* San Diego, CA, USA, 2015.

[74] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, “Deeply-supervised nets," in *Proc. Int. Conf. Artif. Intell. Statisti.,* San Diego, CA, USA, 2015, pp. 562-570.

[75] Y. Deng, Y. Sun, Y. Zhu, M. Zhu, W. Han, and K. Yuan. (2018). A strategy of MR brain tissue images’ suggestive annotation based on modified U-Net. arXiv e-prints. [Online]. Available: https://ui.adsabs.harvard.edu/abs/2018arXiv180707510D

[76] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the Inception Architecture for Computer Vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.,* Las Vegas, NV, 2016, pp. 2818-2826.

[77] P. Kumar, P. Nagar, C. Arora, and A. Gupta, “U-Segnet: Fully convolutional neural network based automated brain tissue segmentation tool,” in *Proc. IEEE 25th Int. Conf. Image Process. (ICIP),* Athens, Greece, 2018, pp. 3503-3507.

[78] V. Badrinarayanan, A. Kendall, and R. Cipolla, “SegNet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.,* vol. 39, no. 12, pp. 2481-2495, Dec. 2017, DOI: 10.1109/TPAMI.2016.2644615.

[79] S. Jégou, M. Drozdzal, D. Vazquez, A. Romero, and Y. Bengio, “The one hundred layers tiramisu: Fully convolutional DenseNets for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops,* Honolulu, HI, USA, 2017, pp. 1175-1183.

[80] R. Basnet, M. O. Ahmad, and M.N.S. Swamy, “A deep dense residual network with reduced parameters for volumetric brain tissue segmentation from MR images," submitted for journal publication.

[81] K. He, X. Zhang, S. Ren, and J. Sun, “Identity mappings in deep residual networks,” in *European Conf. Comput. Vis.,* Amsterdam, Netherlands, 2016, pp. 630-645.

[82] T. Rohlfing, “Image similarity and tissue overlaps as surrogates for image registration accuracy: Widely used but unreliable,” *IEEE Trans. Med. Imag.,* vol. 31, no. 2, pp. 153-163, Feb. 2012, DOI: 10.1109/TMI.2011.2163944.

[83] L. Wang *et al.,* “Benchmark on automatic six-month-old infant brain segmentation algorithms: The iSeg-2017 challenge,” *IEEE Trans. Med. Imag.,* vol. 38, no. 9, pp. 2219-2230, Sept. 2019, DOI: 10.1109/TMI.2019.2901712.

[84] X. Glorot, and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks," in *Proc. Int. Conf. Artif. Intell. Stat.,* Sardinia, Italy, 2010, pp. 249-256.

[85] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification,” in *Proc. IEEE Int. Conf. Comput. Vis.,* Santiago, Chile, 2015, pp. 1026-1034.

[86] C. H. Sudre, W. Li, T. Vercauteren, S. Ourselin, and M. J. Cardoso, “Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations,” in *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support,* Québec City, QC, Canada, 2017, pp. 240-248.

[87] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, and J. Liang, “UNet++: A nested U-Net architecture for medical image segmentation,” in *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support,* Granada, Spain, 2018, pp. 3-11.

[88] A. Paszke *et al.,* “Automatic differentiation in PyTorch," in *Proc. Conf. Neural Inf. Process. Syst.,* Long Beach, CA, USA, 2017.

[89] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge, “Comparing images using the Hausdorff distance,” *IEEE Trans. Pattern Anal. Mach. Intell.,* vol. 15, no. 9, pp. 850-863, Sept. 1993, DOI: 10.1109/34.232073.

[90] Y. Sun, K. Gao, Z. Wu, Z. Lei, Y. Wei, J. Ma, X. Yang, X. Feng, L. Zhao, T. L. Phan, J. Shin, T. Zhong, Y. Zhang, L. Yu, C. Li, R. Basnet, M. O. Ahmad, M.N.S. Swamy, W. Ma, Q. Dou, T. D. Bui, I. H. Gotlib, S. Shultz, L. Li, G. Li, W. Lin, D. Shen, and L. Wang, “Benchmark on multiple site infant brain segmentation algorithms: The iSeg-2019 challenge," under review by *IEEE Trans. Med. Imag.*